UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

The CircularApp: determinando si un envase va al contenedor amarillo

Autor/es

LORENA CASTELLANOS CACHO

Director/es

JÓNATAN HERAS VICENTE

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2017-18

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

The CircularApp: determinando si un envase va al contenedor amarillo

Realizado por:

Lorena Castellanos Cacho

Tutelado por:

Jónathan Heras Vicente

Logroño, febrero, 2018

Índice

Resumen

La Inteligencia Artificial se ha popularizado mucho en los últimos años, ya que la cantidad de mejoras e innovaciones que supone en el día a día de las personas son incontables. La empresa Ecoembes se ha dado cuenta de ello, y lo utiliza para el beneficio del planeta, ya que aplicaciones de este tipo pueden suponer una ayuda notable para las personas que tengan cualquier duda sobre el reciclaje.

Este proyecto va enfocado en esta línea de trabajo. Dentro de las dudas sobre el reciclaje, las más frecuentes son las relacionadas con el contenedor amarillo. Por ello, entre otros proyectos, Ecoembes quería desarrollar una aplicación móvil para que, a partir de la imagen de un producto, determine si va al contenedor amarillo o no. Éste ha sido el problema abordado en este TFG y para ello ha sido necesario el uso de técnicas de tratamiento de imágenes, aprendizaje automático y *deep learning*.

Como primer paso en este Trabajo Fin de Grado se ha llevado a cabo un estudio comparativo de alternativas para elegir el modelo de predicción que genera los mejores resultados. Dicho modelo ha sido desplegado en un servidor web y ha sido la base para desarrollar una aplicación móvil Android. El resultado final es una aplicación móvil completamente funcional capaz de, a partir de la imagen de un envase, predecir con una precisión del 97% si dicho envase va al contenedor amarillo o no.

Abstract

Artificial Intelligence has become very popular in the last years, since it has countless applications in the daily lives of people. The company Ecoembes has realized this, and uses it for the benefit of the planet, because applications of this type can be a significant help for people who have doubts about recycling.

This project is focused on this line of work. In the doubts about recycling, the most frequent are those related to the yellow container. Therefore, among other projects, Ecoembes wanted to develop a mobile application that, based on the image of a product, determines whether it goes to the yellow container. This has been the problem tackled in this work using image processing, machine learning and deep learning techniques.

The first step of this Final Degree Project is a comparative study of alternatives to choose the best prediction model. This model has been deployed on a web server and has been the basis for developing an Android mobile application. The result is a fully functional mobile application able to predict, from the image of a product, with 97% accuracy whether that product goes to the yellow container.

1. Introducción

1.1. Antecedentes

Ecoembes es una organización que cuida del medio ambiente a través del reciclaje y el ecodiseño de los envases en España. Hacen posible que los envases de plástico, latas y briks (contenedor amarillo) y los envases de cartón y papel (contenedor azul) puedan tener una segunda vida.

Su misión es proporcionar a la sociedad una respuesta colectiva de los agentes económicos ante los temas medioambientales relacionados con el consumo de productos envasados domésticos, logrando el cumplimiento de los objetivos marcados por la Ley, con la mayor eficiencia en el uso de los recursos de la compañía.

Ecoembes nació en 1996 y desde entonces ha evitado la emisión de más de 17,7 millones de toneladas de CO_2 a la atmósfera, reciclando 17,9 toneladas de envases. Esta cifra supone también un ahorro de 7 millones de MWh**,** lo mismo que consumirían los móviles que hay en España durante los próximos 40 años. El 17 de mayo de 2017, se inauguró en Logroño el laboratorio *The Circular Lab*, un proyecto pionero en Europa que aglutina todas las propuestas y soluciones para impulsar la innovación en el ámbito de los envases y su posterior reciclado, todo ello enmarcado bajo el concepto de Economía Circular[1]. Este laboratorio incidirá en todas las fases del ciclo de vida de los envases: desde su concepción, a través del ecodiseño, hasta su reintroducción al ciclo de consumo a través de nuevos productos. *The Circular Lab* es el único centro de innovación en el mundo especializado en este ámbito. Todo ello en un marco de estrecha colaboración de innovación abierta entre empresas, administraciones públicas y ciudadanos. En este entorno es clave el compromiso del ciudadano a la hora de reciclar, y la aplicación de las mejores tecnologías en la gestión de los residuos, que se desarrollan no sólo gracias a la actividad del propio centro, sino también mediante el ecosistema emprendedor.
Desde *The Circular Lab* se trabaja en cuatro áreas de investigación diferenciadas:

- El desarrollo de los envases del futuro.
- La aplicación de la tecnología más puntera a los procesos de recogida, selección y reciclado de envases.
- Fomentar la innovación abierta y colaborativa a través del trabajo en red con nuevos emprendedores y *startups* que trabajen en el ámbito de la Economía Circular.
- El desarrollo de herramientas que faciliten la separación en el hogar y el correcto reciclaje.

Es en esta última línea de investigación donde más entra en juego la tecnología y la innovación. Lo primordial es aclarar al ciudadano todas las dudas posibles acerca del reciclaje y la mejor manera para abordar este problema es con la ayuda de la Inteligencia Artificial. Una prueba de ello es que han desarrollado el bot que podemos observar en la *Figura 1*, que es capaz de reconocer la voz para poder resolver la gran parte de las dudas del ciudadano sobre reciclaje.

[1] La Economía Circular apuesta por cambiar el modo de producción, a fin de lograr que cada producto tenga múltiples ciclos de uso y fabricación, esto es, que los recursos se conviertan en productos, los productos en residuos y los residuos en recursos.

Figura 1: Nao, el primer bot de Inteligencia Artificial que ayuda a reciclar

Otra de las líneas en las que se está trabajando es en el reconocimiento de imágenes. En concreto, se pretende crear un sistema que, a partir de una imagen de un envase, sea capaz de indicar el contenedor donde reciclarlo. Dicho sistema podría ser integrado en el bot. Además, también se está trabajando en el reconocimiento de contenedores, con la capacidad de reconocer los caracteres impresos en los contenedores en las partes superior, inferior y laterales. Este TFG sigue esta línea y surge a raíz de realizar prácticas en la empresa Ecoembes. Durante dichas prácticas, se presentó un problema en esta línea de trabajo (la del desarrollo de herramientas que facilitan la separación en el hogar y el correcto reciclaje) cuya resolución se encontraba en el reconocimiento de imágenes.

Presentación del problema durante las prácticas: El problema original consistía en crear un programa capaz de distinguir ciertas características (material, contenedor de retirada, producto que contiene o contenía) de cualquier envase a partir de una foto. Antes de abordar el problema directamente, se resolvió uno más simple. Éste último era realizar un programa que, dada una imagen de una botella, determina si es de plástico o de vidrio.

En este TFG se aborda un problema más ambicioso, ya que lo realizado en las prácticas era una prueba de concepto, y ahora se pretende desarrollar una aplicación completa que pueda ser usada por todos los ciudadanos.

1.2. Justificación del proyecto

Casi a diario, cuando estamos reciclando, nos preguntamos “¿En qué contenedor tengo que depositar este bote?”.

Por más que nos den folletos, los cuales perdemos fácilmente, muchas veces no estamos seguros de a qué contenedor va cierto producto. Además, cada persona nos dice una cosa diferente cuando preguntamos, y no estamos seguros de a quién hacerle caso. Otras veces, nos da pereza buscar información y acabamos depositándolo en el contenedor que nos parece. ¡Cuántas veces nos gustaría tener a mano algo rápido y fácil que de alguna manera nos resuelva nuestra duda en un minuto!

Con el objetivo de solventar este problema, el cual motivó este TFG, Ecoembes decidió familiarizarse con el reconocimiento de imágenes. De esta manera, si una aplicación móvil rápidamente consiguiera decir al momento a través de la imagen de cierto producto a qué contenedor va dicho producto, la mayoría de los ciudadanos la utilizarían y el reciclaje se beneficiaría de ello.

Como este problema es tan amplio, este TFG va enfocado a un determinado contenedor. Se trata del contenedor amarillo. La mayoría de las dudas vienen dadas por la confusión del reciclaje de ciertos productos que se relacionan con el contenedor amarillo y que no es así (como las perchas o los envoltorios de pastillas) o viceversa (como los briks o el papel albal).

Objetivo: Este TFG consistirá en desarrollar una aplicación Android la cual a través de la imagen de un producto que proporcionará un ciudadano, mostrará a éste si dicho producto se deposita o no en el contenedor amarillo. Por lo que cabe destacar que este TFG consta de dos partes notables: la construcción de un modelo de clasificación para determinar a partir de una imagen nueva si se deposita en el contenedor amarillo o no, y el desarrollo de la aplicación Android que utilice dicho modelo.

2. Planificación

En esta sección se aborda la planificación de este proyecto, la cual consta de la metodología que se va a adoptar para su realización, su ciclo de vida y sus hitos, su estructura de desarrollo de trabajo (EDT), el diccionario de la EDT, el diagrama de Gantt con las fases que se van a llevar a cabo a lo largo de este proyecto y por último, el plan de los riesgos que pueden surgir y la gestión de éstos.

2.1. Metodología

Tras un breve análisis, se ha decidido que la metodología más conveniente debido a la forma de este proyecto es la metodología en cascada, ya que esta metodología se caracteriza por ordenar de manera rigurosa las etapas del ciclo de vida de software (dado que el comienzo de cada etapa debe esperar a la finalización de la etapa anterior) y se ajusta a un proyecto como este, donde los requisitos están bien definidos.

Por lo tanto, cuando en la revisión de una etapa se determine que el proyecto no está listo para pasar a la siguiente etapa, permanecerá en la etapa actual hasta que no esté preparado. En la *Figura 2* podemos ver la estructura del modelo en cascada.

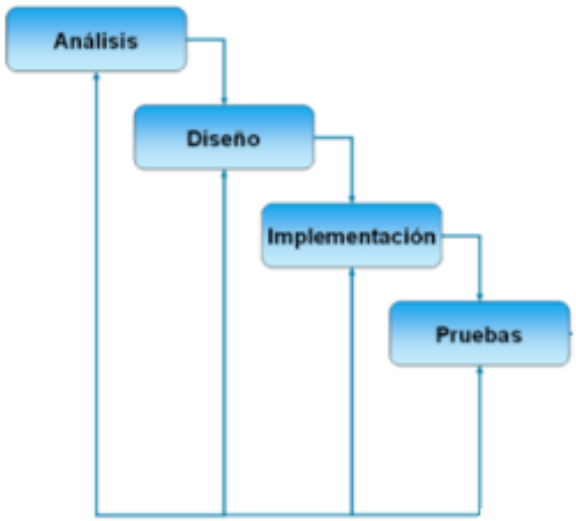


Figura 2: Estructura modelo en cascada

2.2. Calendario e hitos

Podemos observar el ciclo de vida del proyecto y los hitos más relevantes en la *Figura 3*.

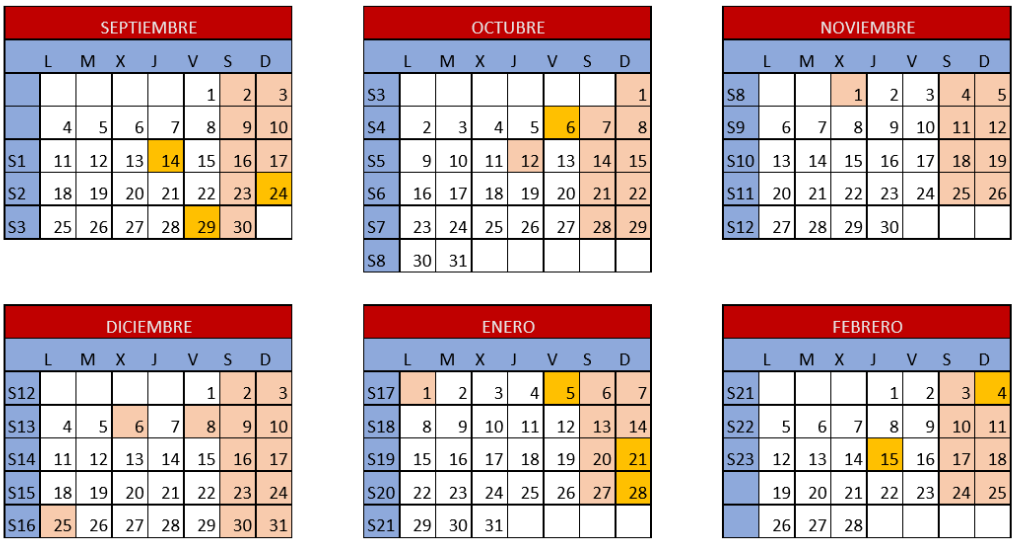


Figura 3: Calendario de los meses que abarca el TFG y los hitos más significativos

Las fechas marcadas en amarillo en la *Figura 3* corresponden a los hitos del proyecto. Concretamente podemos ver a qué hito corresponde cada fecha en la *Tabla 1*. Como se ha explicado en la metodología, la estructura constará de las siguientes fases:

1. Planificación.
2. Análisis.
3. Diseño.
4. Implementación.
5. Pruebas.
6. Seguimiento y control.

Por último, después de realizar estas fases, se obtendrán unas conclusiones. Los hitos que tenemos en la *Tabla 1* corresponden, por tanto, a los cierres de las fases.

Fecha	Año	Hitos
14 de septiembre	2017	Inicio del TFG
24 de septiembre	2017	Cierre de la planificación
29 de septiembre	2017	Cierre del análisis
6 de octubre	2017	Cierre del diseño
5 de enero	2017	Cierre de la implementación
21 de enero	2018	Cierre de las pruebas
28 de enero	2018	Cierre del seguimiento y control
4 de febrero	2018	Cierre de las conclusiones
15 de febrero	2018	Cierre del TFG
13 de marzo	2018	Defensa del TFG

Tabla 1: Tabla de correspondencia fechas e hitos

2.3. Estructura de desarrollo de trabajo (EDT)

En la *Figura 4* se puede ver la estructura de descomposición del proyecto (EDT). Es decir, se muestra la descomposición jerárquica de este proyecto en tareas. En un primer momento, nuestro proyecto se divide en ocho tareas más generales: Planificación, Análisis, Diseño, Implementación, Pruebas, Seguimiento y control, Conclusiones y Memoria. Después, las cuatro primeras se dividen a su vez en tareas más pequeñas (paquetes de trabajo). Todo este detalle puede observarse en la Figura 4.

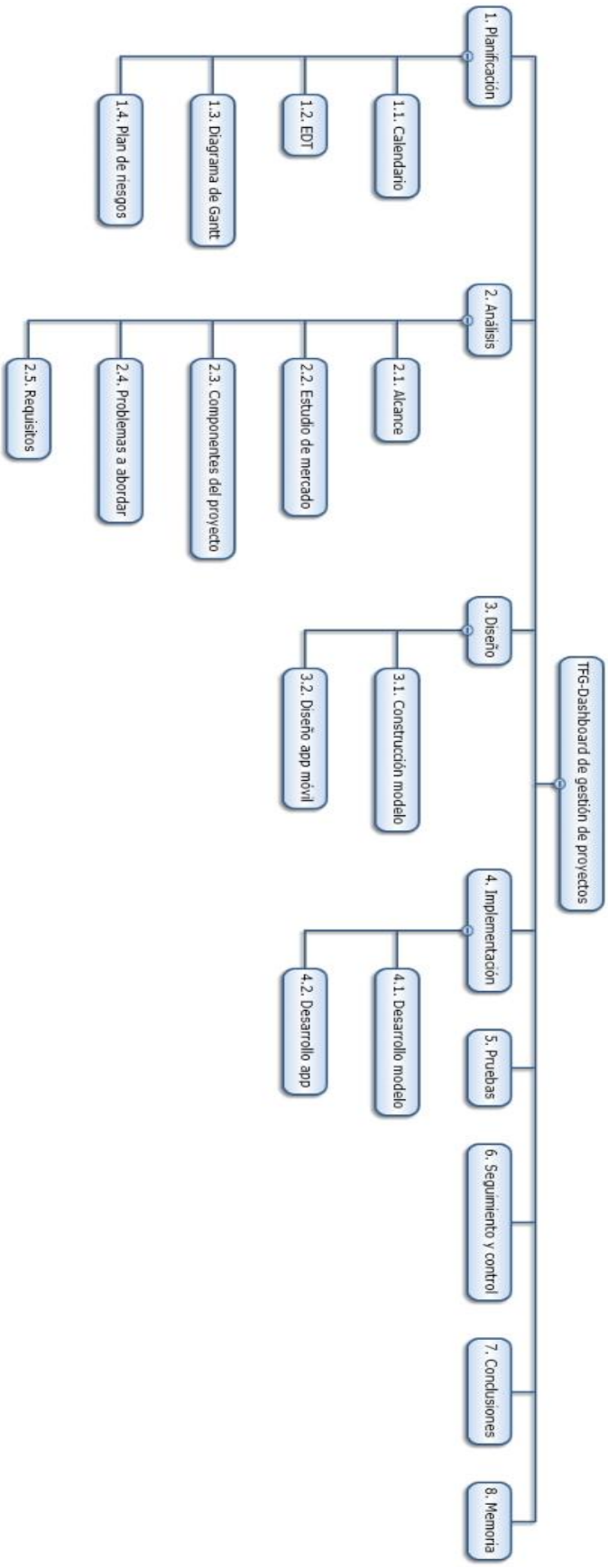


Figura 4: EDT del proyecto

2.4. Diccionario de la EDT

Las tareas de la EDT quedan reflejadas y detalladas en el diccionario de la Tabla 2.

ID	Nombre	Descripción
1	Planificación	
1.1	Calendario	Calendario de los meses de vida del TFG con los hitos y fechas más significativas.
1.2	EDT	Esquema de descomposición de trabajo.
1.3	Diagrama de Gantt	Planificación y programación de tareas, estimadas en horas y duración, a lo largo del TFG.
1.4	Plan de riesgos	Estudio de los posibles riesgos que pueden surgir a lo largo del proyecto, junto con su estrategia de prevención, estrategia de minimización y plan de contingencia.
2	Análisis	
2.1	Alcance	Objetivos que queremos conseguir y diagrama de flujos representativo del proyecto.
2.2	Estudio de mercado	Búsqueda y explicación de aplicaciones similares en el mercado.
2.3	Componentes del proyecto	Explicación de las partes en que se divide el proyecto.
2.4	Problemas a abordar	Requisitos hardware y software del sistema.
2.5	Requisitos	Requisitos funcionales y no funcionales acerca del proyecto.
3	Diseño	
3.1	Construcción modelo	¿Cómo se construye un modelo de clasificación?
3.2	Diseño app móvil	Diseño de alternativas y diseño de interfaz.
4	Implementación	
4.1	Desarrollo modelo	Construcción del dataset, creación de programa y análisis estadístico.
4.2	Desarrollo app	Construcción aplicación móvil.
5	Pruebas	Testeo del desarrollo en un entorno controlado. Comparación de resultados.
6	Seguimiento y control	Seguimiento y control del cumplimiento de la planificación a lo largo del TFG.
7	Conclusiones	Balance del proyecto, lecciones aprendidas y mejoras posibles de implementar.
8	Memoria	Realización de la memoria del TFG a medida que se va desarrollando éste.

Tabla 2: Diccionario de la EDT

2.5. Diagrama de Gantt

Para entender mejor las fases que se llevarán a cabo en este proyecto, puede observarse el diagrama de Gantt de la *Figura 5*. En este diagrama puede verse el desarrollo de las distintas fases, ver gráficamente las tareas de cada una de ellas, su duración y secuencia.

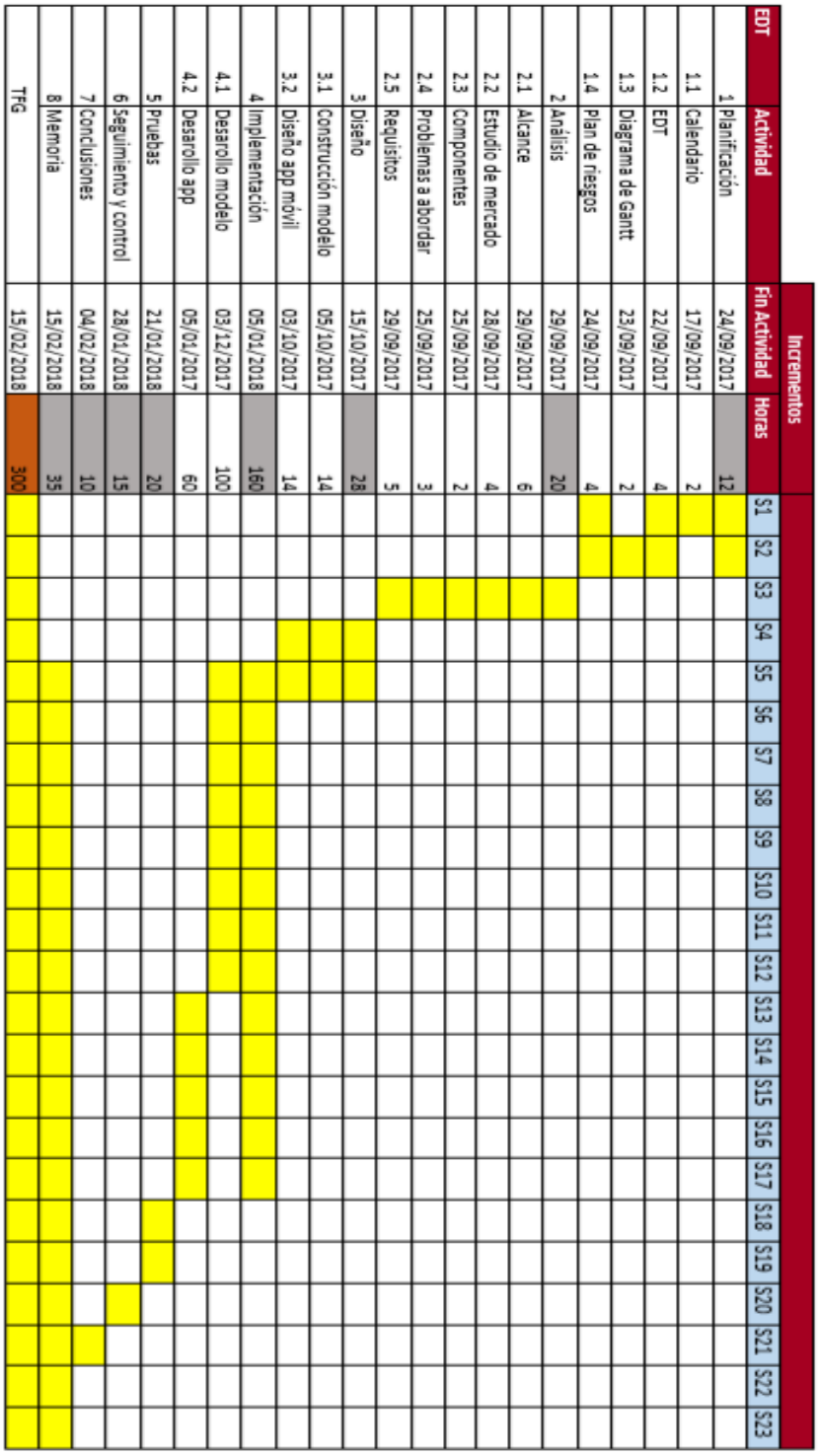


Figura 5: Diagrama de Gantt

2.6. Plan de riesgos

Antes de abordar el plan de riesgos de este proyecto, es necesario conocer los actores involucrados:

- Cliente: El cliente es la empresa Ecoembes. En concreto, quien representa a la empresa en Logroño es Zacarías Torbado Martínez, que es el interlocutor de la empresa.
- Tutor de la Universidad: El tutor de este TFG es Jónathan Heras Vicente, del área de ciencias de la computación e Inteligencia Artificial en la Universidad de La Rioja.

En las siguientes tablas se muestra los posibles riesgos que pueden suceder a lo largo del proyecto en relación a distintos aspectos. En cada uno de los posibles riesgos se describe que es lo que hay que hacer si sucede y que es lo que se debe hacer para evitar que surja ese problema o evitarlo. En la *Tabla 3.1* y en la *Tabla 3.2* podemos ver qué riesgos pueden surgir con el cliente; en la *Tabla 4*, con el tutor; en la *Tabla 5*, con el tamaño del producto; en la *Tabla 6*, con la tecnología y, por último, en la *Tabla 7*, con el desarrollo del producto.

Riesgos con el cliente			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Desaprobación del producto final.	Mantener informado al cliente de los avances para poder cambiarlo si así lo desea el cliente.	Mantener informado al cliente de los avances.	No se puede realizar nada, ya que el proyecto estaría acabado y por tanto, sus horas agotadas.
Ausencia del cliente para resolver dudas del producto.	No se puede prevenir.	Mostrar al cliente dicha posibilidad y sugerir, si no es el caso, que otro empleado de la empresa debería estar al tanto de todo lo relacionado con el producto por si ocurre esto.	Ante la ausencia física del representante del cliente, se le mandará un email, o en su defecto, se le llamará para que resuelva las dudas. Si ninguna de estas opciones es posible, se consultará a otro empleado de la empresa al tanto de los deseos de ésta en cuanto a este producto.
Ausencia de respuesta tras la entrega del producto.	No se puede prevenir	Mantener comunicación con el cliente en la cual quede claro qué le gusta o le desagrada del producto a medida que se va desarrollando.	Intentar comunicarnos con el cliente para que nos de su aprobación o desaprobación acerca del producto.

Tabla 3.1: Riesgos con el cliente

Riesgos con el cliente			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Cambio en los requisitos del proyecto.	Realizar un buen análisis de requisitos y asegurarse de la conformidad por parte del cliente.	Misma estrategia que la de prevención.	Cambiar la planificación del proyecto siempre y cuando se mantengan las horas. Si el cambio en los requisitos superase las horas establecidas, se le comunicará al cliente para que se ajusten según el tiempo indicado.
Cancelación del acuerdo debido a que surge otro producto similar en el mercado.	Realizar un estudio de productos similares en el mercado por si acaso hay alguno similar.	En caso de que un estudio se haya encontrado un producto similar, avisarle al cliente y llegar a un acuerdo con éste para modificar el producto.	Continuar con la realización del proyecto, a pesar de que no se entregue al cliente inicial llegando a un acuerdo con el tutor de la Universidad.
No quedar bien definido el producto de acuerdo con lo que quería el cliente debido a malas interpretaciones.	En cada comunicación con el cliente, preguntarle las dudas hasta que quede claro lo que él quiere. En caso de que no quede claro, ayudarle a definirlo.	Misma estrategia que la de prevención.	Si estamos a tiempo, cambiarlo por lo que realmente quiere el cliente. Hablarlo con él hasta que quede clara la definición del producto.

Tabla 3.2: Riesgos con el cliente

Riesgos con el tutor			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Ausencia del tutor.	No se puede prevenir.	Mantener informado al tutor sobre mi proyecto.	Si es necesario, ponerse en contacto con él, mandarle un correo o llamarle.
Insatisfacción con el avance del proyecto.	Mantener comunicación con el tutor e ir enseñándole los avances, así como preguntarle las dudas generadas hasta que se aclaren.	Quedar semanalmente con el tutor para comentar el progreso del proyecto.	Mejorar el proyecto poniéndose en contacto más a menudo para que pueda dar su opinión acerca de cómo mejorar éste.

Tabla 4: Riesgos con el tutor

Riesgos con el tamaño del producto			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Retrasos del proyecto.	Realizar un estudio para ver la viabilidad del cumplimiento de los requisitos. Dependiendo de éste, modificar los requisitos.	Ir realizando a la vez que el proyecto, el seguimiento y control de éste para comprobar si hay retrasos importantes.	Modificar el alcance del mismo con permiso del cliente o del tutor, en caso de que sea posible y todavía tengamos horas para poder hacerlo.
Disminución de la calidad del proyecto.	Definir con el cliente los criterios de aceptación del producto con respecto a la calidad, y repartir e tiempo de acuerdo a éstos.	Ir realizando una comprobación de la calidad del producto, mostrando éste al cliente para ver si es lo que pedía.	Mejorar la calidad, pero sin pasarnos de tiempo, si es posible, buscando un equilibrio entre ambos.

Tabla 5: Riesgos con el tamaño del producto

Riesgos con la tecnología			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Inexperiencia con la tecnología utilizada.	Planificar en el alcance un tiempo para la formación en esa tecnología, para que luego no se pierda más tiempo del planificado.	Misma estrategia que la de prevención.	Modificar el alcance asignando un tiempo para la formación en esa tecnología, si es posible.
Escoger una tecnología inadecuada.	Análisis y estudio de las tecnologías más convenientes para este proyecto.	Misma estrategia que la de prevención.	Si nos lo permite el tiempo, cambiar de tecnología.

Tabla 6: Riesgos con la tecnología

Riesgos con el desarrollo del producto			
Riesgo	Estrategia de prevención	Estrategia de minimización	Plan de contingencia
Aparición de nuevas tareas no programadas.	Análisis y estudio de nuevas tareas posibles.	Planificar un tiempo en el alcance para la posibilidad de aparición de nuevas tareas.	Modificar el alcance del proyecto si estamos en una fase temprana.
Pérdida de la documentación del proyecto y del código.	Realizar copias periódicas de seguridad en la nube y en un disco externo para la documentación. Para el código usar un repositorio adecuado como GitHub o BitBucket.	No se puede minimizar. Hay que preverlo.	Recuperación de la documentación desde la nube o la restauración de las copias de seguridad. Con respecto al código fuente, recuperación desde el repositorio guardado.
Falta de información para lograr un buen producto.	Estudio de la información necesaria para lograr un producto adecuado.	Recopilar más información de la que se haya estimado necesaria por si surge este problema, no perder tiempo.	Si estamos a tiempo, introducir la información necesaria.

Tabla 7: Riesgos con el desarrollo del producto

3. Análisis

En esta sección se desarrolla el análisis del proyecto y se aborda qué se quiere conseguir al final de éste. Para ello, se realiza un estudio de mercado, se analizan los componentes del proyecto, los problemas que pueden encontrarse y los requisitos, tanto funcionales como no funcionales.

3.1. Alcance

Este proyecto tiene como principal objetivo construir una aplicación móvil que, dada una imagen de un envase, determine si se debe depositar en el contenedor amarillo o no. Esto debe hacerse con una buena precisión. El funcionamiento de la aplicación móvil viene representado por el diagrama de secuencias mostrado en la *Figura 6*.

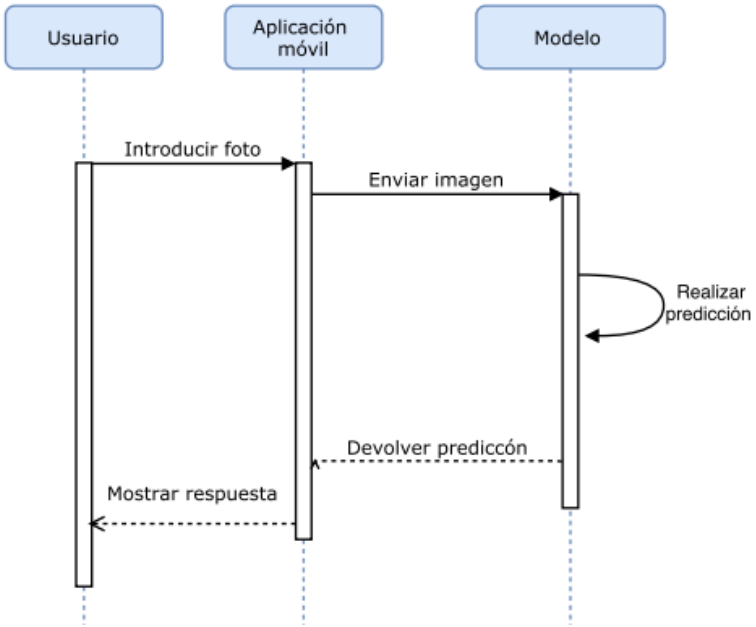


Figura 6: Diagrama de secuencia del proyecto

3.2. Estudio de mercado

Con el objetivo de conocer si hay aplicaciones similares a la que se va a desarrollar, se realiza un estudio de mercado. Para realizarlo, en primer lugar, se han buscado aplicaciones Android, ya que éstas son la competencia, relacionadas con el reciclaje y que sean gratuitas. Para ello, se ha investigado dentro de las aplicaciones que se ofrecen en *Google Play*.

Para llevar a cabo la búsqueda se ha considerado un perfil general. Es decir, aplicaciones que puede usar cualquier ciudadano para resolver ciertas dudas, y no aplicaciones exclusivamente para niños, por ejemplo.

3.2.1. Búsqueda de aplicaciones existentes

Se han encontrado muchas aplicaciones existentes sobre reciclaje. Las cinco más destacadas y las más valoradas por los usuarios son las siguientes:

1. **Separar es reciclar** [1]: Es una aplicación destinada a la gente que le preocupa el medio ambiente patrocinada por el colegio San Agustín de Valladolid. Tiene un buscador y la imagen de cuatro contenedores, como podemos ver en la *Figura 7*. Permite buscar un desecho y muestra al contenedor al que se debe depositar.

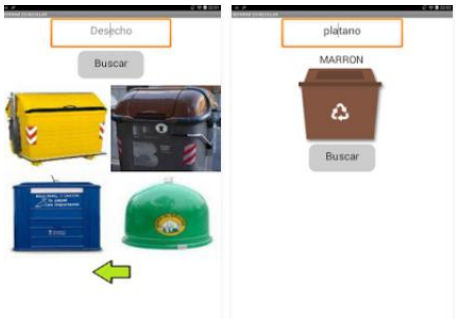


Figura 7: Pantallas de la app móvil "Separar es reciclar"

2. **Guía del reciclaje** [2]: Esta aplicación es de Ecoembes, por lo que es del propio cliente de este proyecto. Ofrece al usuario una guía con toda la información sobre cómo separar correctamente los envases. Incluye un buscador de envases y una descripción de los distintos contenedores, explicando qué residuos se pueden depositar en los mismos y cuáles no, como se puede observar en la *Figura 8*.

Figura 8: Pantallas de la app móvil "Guía del reciclaje"

3. **ReciclApp-BirziklApp** [3]: Esta aplicación se ha realizado en el marco de los convenios que el Gobierno de Navarra mantiene con Ecoembes. La aplicación aclara al usuario qué contenedor debe utilizar según el tipo de residuo, permite localizar puntos limpios geolocalizados (instalaciones de titularidad pública y acceso gratuito para residuos urbanos con características especiales que no deben ser depositados en los contenedores que habitualmente utilizamos en la vía pública) más cercanos, y otros tipos de contenedores como los destinados a aceite, pilas, ropa, zonas de compostaje comunitario e información sobre la recogida de residuos voluminosos; además de otros servicios públicos relacionados con el reciclaje. Se pueden observar algunas de sus pantallas en la *Figura 9*.

Figura 9: Pantallas de la app móvil "ReciclApp-BirziklApp"

4. **Reciclar** [4]: Es una aplicación creada con el objetivo de sociabilizar información sobre qué materiales se pueden reciclar en Tucumán, Argentina, y dónde los reciben. La aplicación fue desarrollada en forma conjunta por la Fundación ProYungas y la Facultad de Bioquímica, Química y Farmacia de la Universidad Nacional de Tucumán. Cuenta con una base de datos georeferenciada de los sitios de recepción, que incluye información como dirección, teléfono, horario de atención, correo electrónico, página web, Facebook. Es bastante completa como se puede ver en la *Figura 10*.

Figura 10: Pantallas de la app móvil "Reciclar"

5. **Aprende a reciclar** [5]: Es un juego para el sistema de reciclado español. Como objetivo tiene enseñar un poco de ecología probando a reciclar muchos objetos de la vida diaria. Cuando se empieza el juego, aparece un objeto en la parte superior derecha de la pantalla que se debe arrastrar hacia el contenedor correcto. Si no se acierta, explica dónde va y en algunas ocasiones, cómo se recicla en España. Se puede ver una pantalla de esta aplicación en la *Figura 11*.

Figura 11: Pantallas de la app móvil "Aprende a reciclar"

Ninguna de las aplicaciones existentes está relacionada con el reconocimiento de envases o productos.

3.2.2. Análisis de la búsqueda

Tras la búsqueda, se ha comprobado que hay diversas aplicaciones sobre reciclaje. De hecho, podemos dividir las aplicaciones encontradas en tres tipos:

1. Aplicaciones informativas (buscadores): Describen las propiedades del artículo, indican dónde se podría reciclar y/o aportan información de otras posibilidades de aplicación tras ser reciclado. Ejemplos: *Separar es reciclar, Guía del reciclaje*, *ReciclApp-BirziklApp*.

2. Aplicaciones lúdicas (juegos): Muestran cómo separar los elementos según correspondan a un contenedor u otro, o explican qué partes dentro de un mismo artículo se deben de separar. Ejemplo: *Aprende a reciclar*.

3. Aplicaciones para la reutilización de materiales (galerías): Selección de imágenes que muestran posibilidades para dar una segunda vida a materiales, reutilizando elementos que de otra forma serían desechos. Algunas indican qué materiales se pueden reciclan y cuáles no. Ejemplo: *Reciclar*.

La aplicación que se desarrollará en este proyecto resolverá dudas, por lo que es meramente informativa. Por otra parte, se centra en el contenedor amarillo. No hay ninguna aplicación que resuelva estas dudas para un contenedor en concreto, excepto del vidrio, lo que beneficia a esta aplicación. Además, algunas de las más populares en el mercado son de Ecoembes o de otras empresas u organizaciones que colaboran con ellos.

El ver que no hay aplicaciones de reconocimiento de imágenes de este tipo, es un punto muy favorable para la aplicación que se va a desarrollar en este proyecto, ya que será la única que relacione la Inteligencia Artificial con la clasificación de productos en el contenedor amarillo. Además, para la gente que ya utiliza las aplicaciones de reciclaje que hay en el mercado actualmente para resolver sus dudas acerca de qué productos se depositan en el contenedor amarillo, será muy útil, ya que les ahorrará tiempo y resolverá esas dudas de una manera cómoda y sencilla (sacando una foto del producto). Por lo que la conclusión es que la aplicación desarrollada en este TFG puede ser exitosa en el mercado.

3.3. Componentes del proyecto

Este proyecto impulsado por la empresa Ecoembes consta de dos partes fundamentales:

1. **Modelo.** Por una parte, será necesario crear un modelo de predicción, con el que se conseguirá identificar los productos a través de las imágenes enviadas por el ciudadano. Es decir, a través de una imagen se identificará si el producto que aparece en ella va al contenedor amarillo o no.

2. **Aplicación móvil.** Por otro lado, tenemos la aplicación móvil. Se tratará de una aplicación Android, la cual permitirá al usuario dos opciones:

    - Pasarle una foto que tenga en la galería de su teléfono.
    - Sacar una foto al momento para enviarla.

    La última opción será la más utilizada, ya que la aplicación está pensada para resolver dudas al momento y que quien la utilice pueda reciclar correctamente. A partir de dicha foto, la aplicación utilizará el modelo del apartado 1 para realizar las predicciones. Es decir, este modelo tendrá de entrada datos nuevos (una nueva imagen en nuestro caso) y como salida, la clasificación de esa imagen según los patrones que se han detectado en el entrenamiento (si va al contenedor amarillo o no).

3.4. Problemas a abordar en el proyecto

Para este proyecto, tenemos que abordar cada parte por separado. Es decir, por una parte la construcción del modelo, y por otra el desarrollo de la aplicación móvil.

- Para el modelo se realizará un estudio de alternativas para ver cuál es la mejor opción. Se utilizarán técnicas de *deep learning* (en concreto, redes convolucionales) y visión por computador.
- Para la aplicación móvil existen dos opciones para realizar las predicciones:

    1. Almacenar el modelo de predicción directamente en la aplicación.
    2. Hacer uso de un servidor.

    Se realizará un estudio para escoger la opción más adecuada para este proyecto.

3.5. Requisitos funcionales y no funcionales

3.5.1. Requisitos funcionales

Requisitos funcionales del modelo:

1. Para el modelo, se construirá un banco de imágenes con algunas de ellas que contengan productos que se depositen en el contenedor amarillo y el resto, con productos que no se depositen en dicho contenedor.
2. El banco de imágenes contendrá alrededor de 200 fotografías en un principio (100 de cada tipo), aunque más adelante podrán añadirse más imágenes. El banco de imágenes ha de estar siempre balanceado (mismo número de imágenes "positivas" y "negativas").
3. El banco de imágenes se dividirá en fotografías que se utilicen para el entrenamiento del modelo, y otras para el test.
4. El modelo será capaz de reconocer imágenes con las siguientes características:
    - El producto es el elemento principal de la imagen.
    - El estado del producto corresponde con el estado en que se va a depositar dicho producto al contenedor, por lo que va a estar algo deteriorado. Es cuando los ciudadanos van a utilizar la aplicación móvil, en el momento en que van a tirar dicho producto. Por lo que es mucho mejor fotografías de este tipo.
    - El producto ha de verse claro.
    - El fondo de la imagen ha de adecuarse con los entornos donde los ciudadanos van a hacer la mayoría de las fotos. En este caso, una cocina, un salón o la calle.
    - La imagen ha de ser capturada con la cámara de un móvil y no una cámara de alta calidad.
5. Los elementos que van al contenedor amarillo se pueden observar en la primera columna de la *Tabla 8*.
6. Los elementos que no van al contenedor amarillo y que reconocerá el modelo como tal, se pueden observar en la segunda columna de la *Tabla 8*.

Objetos que se depositan en el contenedor amarillo	Objetos que no se depositan en el contenedor amarillo
Latas Briks Papel Albal Yogures Botellas de plástico	Perchas Cepillos de dientes Envoltorios de pastillas Botellas de vidrio Cajas de cartón

Tabla 8: Clasificación de productos respecto al contenedor amarillo

Requisitos funcionales de la aplicación móvil:

1. La aplicación será exclusivamente para el móvil, no habrá ninguna para el PC.
2. La aplicación debe ser compatible con las nuevas versiones del sistema operativo Android.
3. La aplicación constará de dos opciones: una que permita subir una foto que esté almacenada en la galería del teléfono móvil, y otra opción que permita sacar una foto.
4. Si se marca la opción de subir foto desde galería, se deberá seleccionar una imagen de las ya almacenadas en el dispositivo. Se entenderá que dicha imagen es la que desea analizar.
5. Si se marca la opción de sacar foto, se mostrará la cámara para que se pueda sacar una foto. Habrá una opción de "Comprobar" para que sea dicha imagen la que se analice.
6. Si se saca una foto y no se quiere enviar porque se desea hacer otra, se podrá volver atrás.
7. La aplicación visualizará una respuesta en la pantalla, una vez capturada la imagen.
8. No será necesario estar registrado para acceder a la App.
9. La aplicación tendrá los logos de *The Circular Lab* y de Ecoembes. Ambos tendrán el mismo tamaño.

En la *Figura 12* podemos ver el diagrama de actividad del uso de la aplicación móvil cuando el usuario desea comprobar si un producto se deposita en el contenedor amarillo o no, sacando la foto a éste desde la cámara o seleccionándola desde galería.

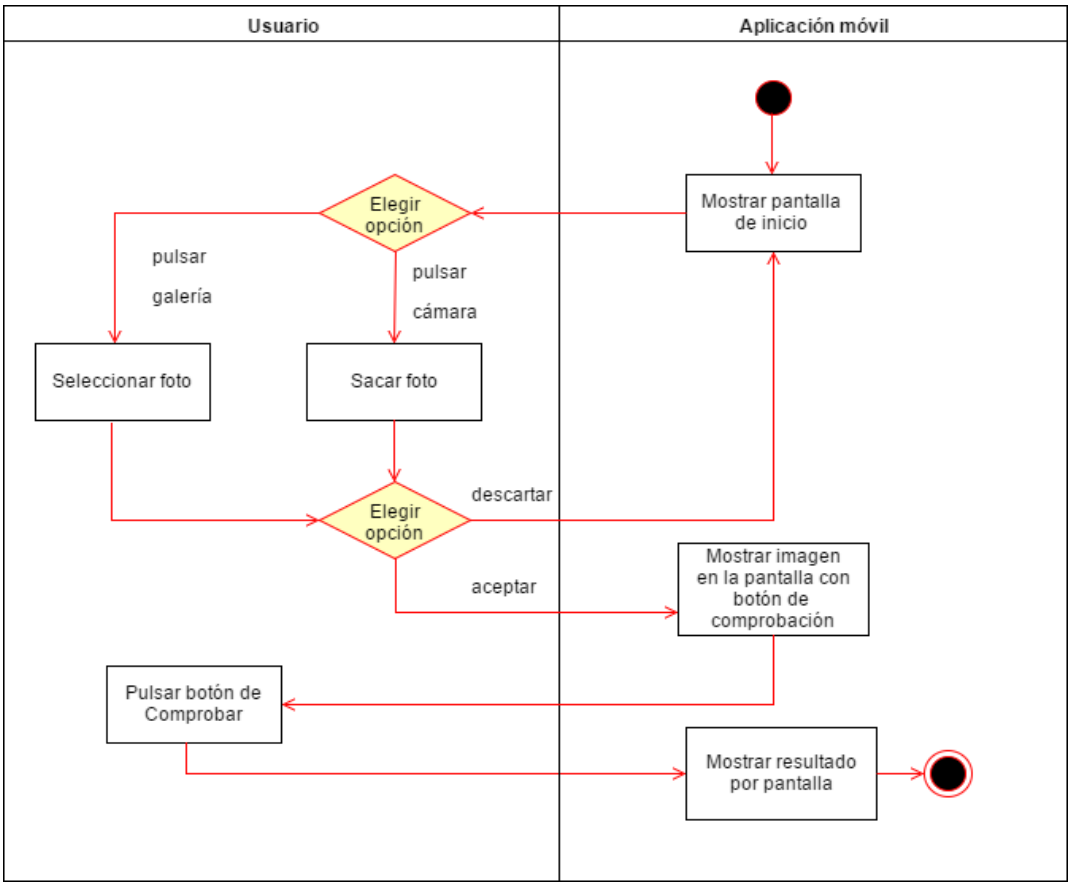


Figura 12: Diagrama de actividad del uso de la aplicación móvil

3.5.2. Requisitos no funcionales

Requisitos no funcionales del modelo:

1. El banco de imágenes será creado con fotos obtenidas del día a día.
2. Estudio de diferentes alternativas para el modelo mediante análisis estadístico.
3. La tasa de acierto ha de ser mayor al 75%.
4. Se analizará cómo será accesible el modelo.

Requisitos no funcionales de la aplicación móvil

1. La aplicación necesita que se haga uso de una cámara de smartphone con un mínimo de 3 Mega píxeles.
2. La aplicación debe ser fácil de utilizar. Para ello nos basaremos en un diseño básico e intuitivo.
3. La aplicación debe de proporcionar tiempos de respuesta rápidos.
4. La aplicación debe ser fácil de analizar y modificar para corregir posibles fallos.
5. La aplicación debe informar al usuario de los permisos que necesita.
6. La temática y el estilo de la aplicación seguirá el diseño de Ecoembes.

4. Diseño

Para entender mejor este apartado, es necesario entender algunos conceptos previos. Por este motivo, se explican las nociones necesarias a continuación. Además, la sección del diseño del proyecto consta de dos importantes apartados. El primero de ellos es el diseño del modelo, y el segundo el diseño de la aplicación móvil.

4.1. Nociones previas

Aprendizaje Automático
El aprendizaje automático [6] es la rama de la Inteligencia Artificial [7] que se dedica al estudio de los agentes/programas que aprenden o evolucionan basados en su experiencia, para realizar una tarea determinada cada vez mejor. El objetivo principal de todo proceso de aprendizaje es utilizar la evidencia conocida para poder crear una hipótesis y poder dar una respuesta a nuevas situaciones no conocidas.

Aplicaciones del aprendizaje automático
Hoy en día existen aplicaciones que utilizan agentes basados en aprendizaje en numerosas ramas de la industria y de la ciencia. Por ejemplo: procesamiento del lenguaje natural, sistemas de recuperación de información, diagnósticos médicos, ciencias biológicas, finanzas e industria bancaria, análisis de imágenes, juegos, robótica y sistemas de recomendación.

Tipos de aprendizaje automático
Existen diversos tipos de aprendizaje automático, siendo los más relevantes los listados a continuación:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semisupervisado
- Aprendizaje por refuerzo
- Transducción
- Aprendizaje multi-tarea

En este proyecto, se usará el aprendizaje supervisado. Para ver una breve explicación de los demás tipos de aprendizaje, se puede consultar el Apéndice 1.

El **aprendizaje supervisado** [6] es una técnica para deducir una función de predicción a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos: una componente del par son los datos de entrada y el otro, el resultado deseado para esos datos. Por ejemplo, el caso que se aborda en este TFG se ajusta muy bien al aprendizaje supervisado, ya que, a partir de un objeto de entrada, en este caso, una fotografía de un envase, tiene que predecir su valor correspondiente, es decir, si va al contenedor amarillo o no. La salida de la función de predicción puede ser un valor numérico (como en los problemas de regresión) o una etiqueta de clase (como en los de clasificación). El objetivo del aprendizaje supervisado es el de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada, válida después de haber visto una serie de ejemplos, los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a las situaciones no vistas previamente. A dicha función se la conoce como modelo de predicción o clasificación.

4.2. Construcción del modelo

La clasificación de imágenes es una tarea común en visión por computador para la cual se suele construir un modelo de predicción que consta de los pasos mostrados en la *Figura 13*.

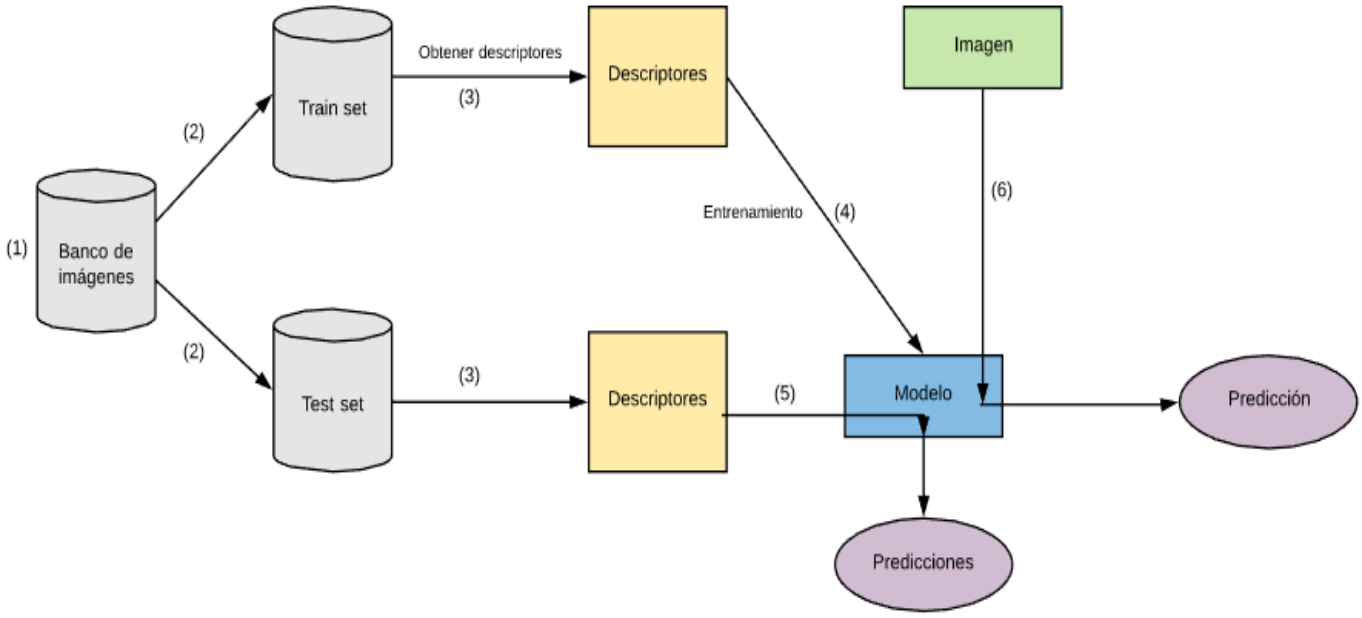


Figura 13: Proceso para la construcción del modelo

A continuación, explicamos en qué consisten los pasos que podemos ver en la *Figura 13*. Posteriormente, serán más detallados con nuestro problema concreto.

1. Recolección de las imágenes
El primer paso para la construcción del modelo es la recolección de datos (construcción del dataset) para crear el banco de imágenes. Este banco deberá estar formado por la misma cantidad de imágenes de cada clase. Esto es muy importante para evitar sesgos. Es decir, si hay más imágenes de una clase que de otra, habrá más posibilidad de que, dada una nueva instancia, que es un ejemplo individual e independiente del objeto que se quiere aprender, la relacione más con la clase más frecuente. Con estas imágenes, se pretende conseguir una representación del "mundo real", es decir, de lo que el modelo se puede encontrar en el día a día.

2. División del dataset
Tras la recolección de imágenes, viene la división del dataset para evaluar la calidad del modelo. Es decir, tendremos que dividir el dataset en dos conjuntos:

- El conjunto de entrenamiento, que se utiliza para entrenar a los algoritmos que darán lugar a distintos modelos.
- El conjunto de test, que se utiliza para evaluar el rendimiento de los algoritmos. Sin embargo, este conjunto no es utilizado para decidir acerca del algoritmo o sobre qué parámetros usar dentro del mismo, ya que carece de sentido porque estima cómo de bien se entrena el modelo, y en caso contrario, sería como "hacer trampas".

3. Extracción de descriptores
El sistema necesita medir los valores que dan una serie de características o atributos de esa imagen. Estas características son los llamados descriptores y serán utilizados para entrenar el modelo. Tanto en el conjunto de entrenamiento, como en el de test, los descriptores deben ser los mismos, es decir, se estudian para todas las imágenes las mismas características, aunque, obviamente, los valores de esas características para cada imagen serán distintos que para el resto.

4. Entrenamiento del algoritmo
En esta etapa comenzamos a utilizar las técnicas de aprendizaje supervisado. Este paso consiste en entrenar distintos algoritmos de aprendizaje para extraer información y poder clasificar las imágenes que le pasamos. Cada uno de estos algoritmos entrenados da lugar a un modelo de predicción.

5. Evaluación de los modelos
Una vez entrenados los modelos, se prueban para ver cuál funciona mejor en cada problema concreto. Para evaluar los modelos se utiliza el conjunto de test. Cabe mencionar el *No free lunch theorem* [8], el cual dice que no existe un algoritmo de clasificación que sea el mejor para todos los problemas, es por esto por lo que es necesario probar distintos algoritmos.

6. Predicción de nuevas imágenes
Una vez el modelo ha sido entrenado, puede utilizarse para predecir la categoría de nuevas imágenes.

Cada uno de estos pasos conlleva un estudio de alternativas, como se puede observar a continuación junto con la explicación de las mismas.

4.2.1. Alternativas para la construcción del banco de imágenes

El primer paso para cualquier aplicación que utilice aprendizaje automático, como es el caso de esta aplicación, es la captura de instancias que formen nuestro banco de datos o dataset. Para la construcción del dataset existen dos opciones en el caso de este proyecto. Estas opciones son las siguientes:

1. **Descargar imágenes de Internet.** Las imágenes pueden ser descargadas de algunos sitios web. Hay numerosos, pero preferentemente se han buscado gratuitos como pueden ser los siguientes:
    - **Google** [9]**:** Las imágenes pueden ser descargadas de Google siempre y cuando la licencia nos lo permita. Es decir, que la licencia sea de dominio público Creative Commons 0 (CC0).
    - **ImageNet** [10]**:** Es una de las mayores bases de datos de imágenes del mundo. Concretamente, fue creada con el objetivo de ayudar a las inteligencias artificiales a entender lo que ven. En la actualidad, es utilizada la mayoría de las veces por los investigadores a la hora de enseñar a los algoritmos de Inteligencia Artificial a reconocer imágenes. De hecho, esta base de datos se utilizó durante las prácticas en Ecoembes para descargar imágenes sobre botellas (de plástico y de vidrio).
    - **Pexels** [11]**:** Es un banco de imágenes gratuito, con una amplia comunidad de fotógrafos que le permiten sumar cinco fotografías de alta calidad todos los días. También cuenta con fotografías de pago.

- **Pixabay** [12]**:** Es uno de los pocos bancos de imágenes en castellano. Posee un buscador con varios filtros.
- **Unplash** [13]**:** Uno de los bancos de imágenes CC0 más completos que existen. Además de contener fotos de calidad excelente sin costo alguno, está en constante crecimiento manteniendo la calidad de las fotografías. Permite hacer clic en los autores y tener un listado de todas las imágenes que han cargado, lo que es particularmente útil cuando se busca un estilo específico, o imágenes de un mismo tema o formato.

2. **Capturar las fotos utilizando una cámara de móvil.** Las fotografías pueden ser sacadas desde un móvil, una tablet o una cámara de fotos. Estas imágenes pueden ser hechas por:

    - **Un único usuario**.
    - **Múltiples usuarios** que envíen las fotografías a una dirección específica para que puedan ser todas reunidas.

El objetivo de este TFG es conseguir una aplicación real que funcione con imágenes con las características descritas en los requisitos (ver sección 3.5.1 del apartado *3. Análisis*). Los modelos de clasificación son susceptibles a la distribución de las imágenes con las que son entrenadas, por lo que conviene que las imágenes usadas para entrenar el modelo, tengan las mismas características que las imágenes que se usarán luego en la aplicación real. Por ello, se capturarán las imágenes utilizando la cámara de un móvil, dado que los ciudadanos sacarán las fotos para la aplicación con el móvil. En este caso, lo harán múltiples usuarios, ya que es la mejor opción por lo comentado anteriormente. En concreto, lo harán diez trabajadores de la empresa, entre los que se incluyen la desarrolladora de este proyecto. Al sacar las imágenes se deberán mandar a un número de WhatsApp ya especificado. De esta forma, se pueden obtener imágenes con las características que buscamos sin necesidad de recorrer muchas fotografías en Internet hasta encontrar las que necesitamos.

4.2.2. Dataset Split

El objetivo cuando se aplican algoritmos de aprendizaje automático es que dichos algoritmos sean capaces de generalizar. Por generalización se entiende el cómo de bien se aplican los conceptos aprendidos por un modelo de aprendizaje automático, a nuevos ejemplos que el modelo no ha visto mientras se entrenaba. Para conseguir esto, es necesario dividir el dataset, lo que se puede hacer mediante los siguientes métodos:

1. Método de *holdout* [14].
2. Método de validación cruzada o *cross-validation* [15].

Se explica cada uno de ellos a continuación.

1. **Método de *holdout***

En este caso, se divide el dataset en un conjunto de **entrenamiento**, que se usa por uno o más métodos para crear clasificadores, un conjunto de **validación**, para optimizar los parámetros de esos clasificadores, y un conjunto de **test**, para calcular el ratio de error del clasificador. Para realizar esta división, hay algunas alternativas. Sin embargo, las más comunes son las que aparecen en la *Figura 14*.

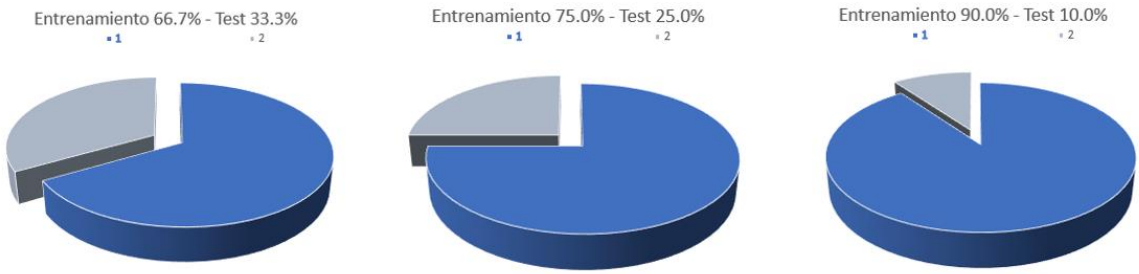


Figura 14: Alternativas para la división del dataset en conjunto de Entrenamiento (azul más oscuro) y conjunto de Test (azul claro)

A su vez, se extraerá del conjunto de entrenamiento, un nuevo conjunto que será el de validación. Para éste, se utilizará entre un 10% y un 20% del conjunto de entrenamiento. Este conjunto evita que utilicemos las imágenes entrenadas para la validación. De esta manera, encontraremos qué parámetros de los algoritmos de aprendizaje producen mejores resultados sin caer en un error.

2. Método *cross-validation*

La idea principal del método de *cross-validation* es la división del conjunto de datos en diferentes subconjuntos. Este procedimiento se basa en repetir el entreno y la evaluación de un modelo usando en cada repetición subconjuntos de entreno y test distintos.

Aunque existen diversas variantes de este método, la más utilizada es la *k-fold cross validation*. Este método se basa en dividir la base de datos en k partes iguales, a las que llamamos *folds* (o partes). En cada iteración se usan k-1 *folds* para entreno y 1 para validar, de modo que se entrenan y evalúan k clasificadores distintos. El hecho de tener que realizar k ejecuciones supone un aumento del coste computacional, pero tiene ventajas frente al método de *holdout*, ya que permite realizar un análisis estadístico, lo que ofrece más seguridad que el método de *holdout*, el cual puede dar buenos resultados por casualidad (tomando justamente una partición determinada). El resultado final de la evaluación se obtiene promediando los resultados de las distintas repeticiones. En la *Figura 15* se puede observar la división del dataset utilizando este método.

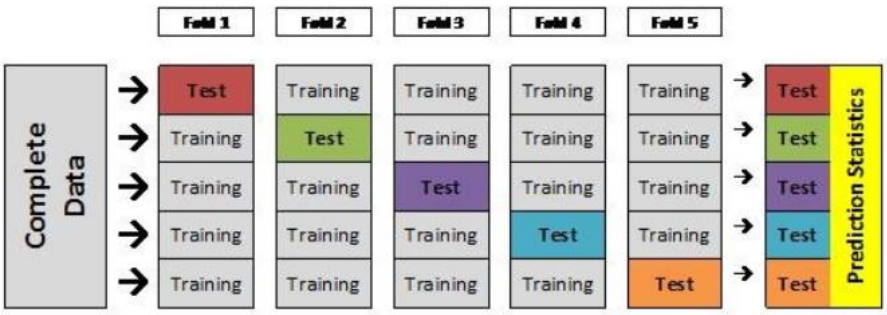


Figura 15: División datos usando 5-fold cross-validation

El número de *folds* en que se debe dividir la base de datos depende de la cantidad de datos de los que dispongamos. Con un número elevado de *folds*, el sesgo del estimador del error será pequeño pero su varianza elevada. Así mismo, el coste computacional será mayor. Con un número pequeño de *folds*, el sesgo del estimador del error será elevado y su varianza pequeña. El coste computacional será inferior. Normalmante, suele utilizarse con k=10, es decir, se suele dividir la base de datos en 10 *folds*.

En este proyecto, se optará por el método de *cross-validation*, ya que a pesar de ser más costoso, permite hacer un análisis más preciso.

4.2.3. Extracción de descriptores

La extracción de los descriptores puede ser un proceso costoso. Hay que tener en cuenta que los descriptores pueden tomar distintos valores para cada imagen. Esta etapa es muy importante, ya que las características generadas serán usadas en el proceso de clasificación de los datos. Así que, a partir de cada instancia de nuestro dataset, debemos obtener un conjunto de propiedades, representado mediante un vector de valores, que describan dicha instancia, teniendo en cuenta que no todas las características son igual de significativas.

¿Cómo deberían ser los descriptores?
Queda la cuestión de qué descriptores usar. Los descriptores deberían ser:

- **Discriminativos**: Objetos similares tendrán valores similares, y objetos diferentes tendrán valores diferentes. Es decir, dos fotografías de botellas de plástico (que van al contenedor amarillo), darán valores similares.
- **Invariantes** a ciertas transformaciones dependiendo del contexto. Por ejemplo, una imagen de una percha y una imagen que sea la rotación de la primera, tienen que dar valores similares.
- **Robustos a cambios** en el entorno, lo que también depende del contexto. Es decir, con una imagen de un brik de leche en una cocina, y otra con un brik de leche en un salón, se deberá tener buenos resultados para ambas.

Hay diversos descriptores para la clasificación de imágenes, pero antes, se debe distinguir entre dos modos de extracción de descriptores:

1. **Modo clásico**. Una persona determina qué características se van a estudiar en cada imagen. Es decir, el programador tiene que ser muy específico al decirle al ordenador qué tipos de cosas debe buscar para decidir si una imagen contiene un determinado objeto o no lo contiene. Es un proceso laborioso y la tasa de éxito depende totalmente de la capacidad del programador para definir con precisión un conjunto de características lo suficientemente descriptivas y discriminativas para el objeto. Ejemplos de métodos clásicos para la extracción de descriptores son los histogramas de color, los histogramas Haar y los histogramas de gradientes orientados. Se puede encontrar una breve descripción de cada uno de los tres en el Apéndice 2.

2. Modo transfer learning. Para hablar de este modo, es necesario conocer algunos conceptos que se exponen a continuación.

- **Redes neuronales artificiales** [16]**:** Es uno de los algoritmos de aprendizaje automático más utilizado para la clasificación. Las redes artificiales de neuronas tratan, en cierto modo, de replicar el comportamiento del cerebro, donde tenemos millones de neuronas que se interconectan en red para enviarse mensajes unas a otras. Esta réplica del funcionamiento del cerebro humano es uno de los "modelos de moda" por las habilidades cognitivas de razonamiento que adquieren. El problema es que son lentas de entrenar, y necesitan mucha capacidad de cómputo. Quizás sea de los modelos que más ha ganado con la "revolución de los datos"; tanto los datos como materia prima, como procesadores de entrenamiento, le vienen como anillo al dedo para las necesidades que tienen estos algoritmos.

- **Deep learning** [17]: El *deep learning* o aprendizaje profundo es un campo del aprendizaje automático que se encarga de emular el enfoque de aprendizaje que los seres humanos utilizan para obtener ciertos tipos de conocimiento. Frente a la linealidad de los algoritmos tradicionales, los algoritmos de *deep learning*, se apilan en una jerarquía de mayor complejidad y abstracción. Cada nivel de abstracción se crea con el conocimiento que se obtuvo de la capa precedente de la jerarquía. Cada nivel en la jerarquía aplica una transformación no lineal en su entrada y utiliza lo que aprende para crear un modelo estadístico como salida. Las iteraciones continúan hasta que la salida ha alcanzado un nivel de precisión aceptable. El número de capas de procesamiento a través de las cuales los datos deben pasar es lo que inspiró la etiqueta de profundidad ("*deep*").

  El proceso de entrenar redes *deep* es muy costoso, necesita muchas imágenes y una gran capacidad de cómputo. Por este motivo, es más valioso utilizar el *transfer learning* para nuestro proyecto, el cual se explica a continuación.

- **Transfer learning** [18]**:** El *transfer learning* consiste en aprovechar el conocimiento adquirido para resolver un problema y usarlo para resolver otro problema similar. De este modo, el aprendizaje es útil y productivo, ya que puede ser aplicado en diferentes situaciones sin necesidad de comenzar de nuevo el proceso. Existen diversas redes neuronales profundas (VGG16, Resnet, …) que han sido entrenadas para clasificar una gran variedad de objetos y que pueden ser usadas para realizar *transfer learning*. Estas redes reciben como entrada imágenes sin ningún tipo de procesado y van aprendiendo en cada nivel de la red características de menor a mayor complejidad que sirven para clasificar los objetos. El modo *transfer learning* elimina la última capa de esas redes y utiliza la salida de las mismas como extractor de descriptores.

Por lo que en el modo *transfer learning*, los descriptores son aprendidos por el ordenador al resolver una tarea, mientras que en el modo clásico, los descriptores son diseñados por personas. Esta diferencia se puede apreciar en la *Figura 16*.

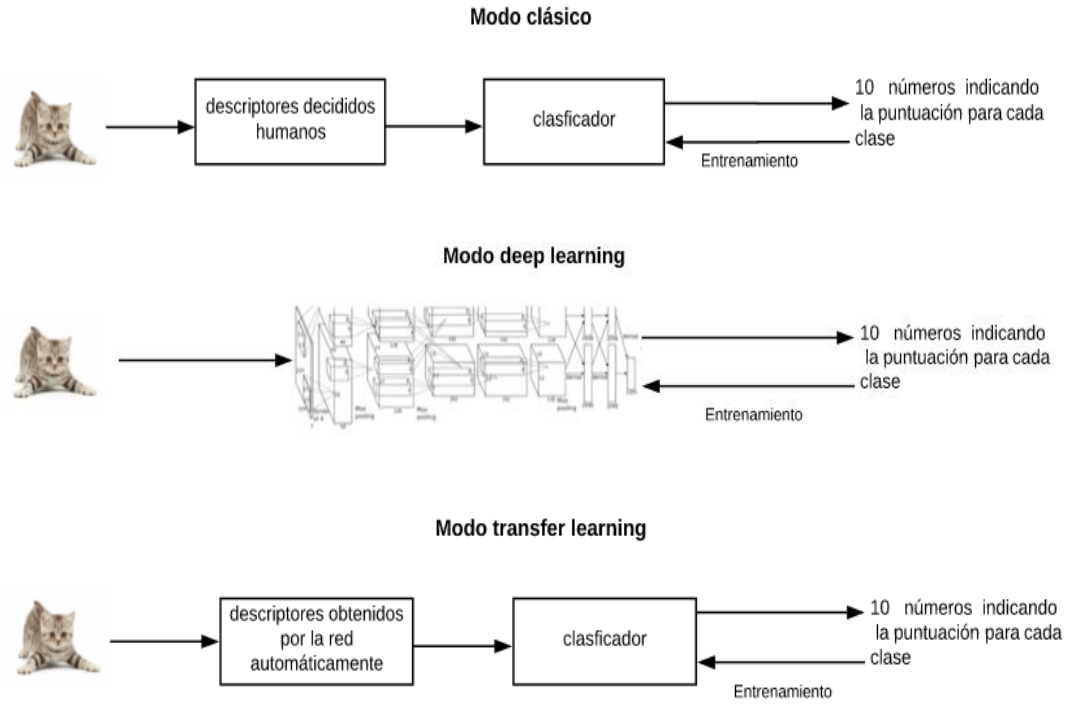


Figura 16: Diferencia entre modo clásico, modo deep learning y modo transfer learning

Extraer descriptores de imágenes de productos que van al contenedor amarillo y que no van, puede ser un trabajo muy tedioso, y es mucho más factible si usamos descriptores que ya hayan sido utilizados para resolver otros problemas de manera exitosa. Por este motivo, entre estos dos modos de extracción de descriptores, se ha escogido el modo *transfer learning*, ya que se adapta mucho mejor a nuestro proyecto.

En este proyecto, se van a estudiar una serie de redes para extraer descriptores con el fin de, posteriormente, analizar los resultados. En concreto, son las redes entrenadas para resolver el reto Imagenet [19], que consiste en un desafío de reconocimiento de imágenes que evalúa los algoritmos para la detección de objetos y la clasificación de imágenes a gran escala. Explicar estas redes en detalle está fuera del alcance de este proyecto, por lo que en la *Tabla 9* se da únicamente el año de publicación, el número de capas y la tasa de error de cada una de ellas para el reto Imagenet.

Redes	Año de publicación	Número de capas	Tasa de error
DenseNet [20]	2016	140	3,74%
GoogleNet [21]	2014	22	6,7%.
Inception [21]	2015	48	8,41%
Xception [22]	2017	36	6,78%
Overfeat [23]	2013	8	13,6%
ResNet [20]	2015	50	3,6%
VGG16 (VGG19) [24]	2014	16 (19)	7,3%

Tabla 9: Características de una serie de redes

Para este proyecto, tras realizar un estudio estadístico (se explicará más adelante), se utilizará aquella red para la cual se obtenga una mejor predicción en nuestro problema concreto.

4.2.4. Algoritmos de clasificación

Tras lo visto en los puntos anteriores, viene la construcción de un modelo que nos permita clasificar nuevas instancias. Las consideraciones para un clasificador son las siguientes:

- Exactitud: proporción de clasificaciones correctas.
- Rapidez: tiempo que toma hacer la clasificación.
- Claridad: cómo de comprensible es para los humanos.
- Tiempo de aprendizaje: tiempo para obtener o ajustar el clasificador a partir de datos.

Algoritmos de clasificación
Algunos de los algoritmos de clasificación más utilizados en el aprendizaje automático, y que son los que serán utilizados para construir modelos de aprendizaje en este trabajo, pueden verse a continuación.

1. **KNN o K vecinos más cercanos** [25]**:** Se fundamenta en una idea muy básica e intuitiva. La idea principal de este algoritmo es clasificar una nueva instancia en la clase más frecuente a la que pertenecen sus k vecinos más cercanos. Su fácil implementación hace que sea un paradigma clasificatorio muy extendido para establecer tasas mínimas de rendimiento.

2. **Árboles de decisión** [26]**:** Un árbol de decisión es un modelo de predicción tal que, dada una base de datos, se construyen diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que suceden de forma sucesiva, para la resolución de un problema.

3. ***Extra trees*** [27]**:** Es un algoritmo de clasificación que se adapta a una serie de árboles de decisión aleatoria (también conocidos como *extra trees*) en varias submuestras del conjunto de datos y utiliza promedios para mejorar la precisión predictiva y controlar el ajuste excesivo.

4. ***Random Forest*** [28]**:** Es una combinación de árboles de decisión tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Construye una larga colección de árboles no correlacionados y luego los promedia.

5. **Regresión logística** [29]**:** En regresión logística la hipótesis es elegida como una función sigmoidea:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T X}}$$

la cual puede escribirse también como la combinación de las siguientes ecuaciones:

$$\begin{aligned} h_\theta(x) &= g(\theta^T X) \\ g(z) &= \frac{1}{1 + e^{-z}} \end{aligned}$$

La función g(z) tiene las siguientes características:

- g(z) ≥ 0,5 si z ≥ 0, lo cual indicará la etiqueta "1"
- g(z) ≤ 0,5 si z ≤ 0, lo cual indicará la etiqueta "0"

Se define la frontera de decisión por aquellos puntos que cumplen que hθ (x) = 0,5. La frontera de decisión es una propiedad de la función de hipótesis, y no de los datos de entrenamiento. De esta manera la función de coste es:

$$J(\theta) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i log(h_\theta(x^i)) + (1-y^i) log(1-h_\theta(x^i))\right)$$

La función de coste J(θ) está compuesta por dos términos, lo cuales se anulan para cada una de las etiquetas: 0 ó 1. El objetivo, por lo tanto, es minimizar la función de coste para entrenar el algoritmo.

6. **SVM** [30]**:** Las máquinas de soporte vectorial, máquinas de vectores de soporte o máquinas de vector soporte (Support Vector Machines, SVM) son un conjunto de algoritmos de aprendizaje supervisado. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase. Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

En el análisis estadístico, se prueba, con cada descriptor, todos estos algoritmos de clasificación. Una vez que se haya hecho el análisis, se escoge el algoritmo de clasificación entre éstos dependiendo de los resultados.

4.2.5. Evaluación

Un concepto fundamental en el ámbito de los métodos de aprendizaje automático son los diversos criterios para la evaluación de los clasificadores. Es decir, estimar la bondad de un clasificador. A esto se conoce como proceso de validación, y nos permite efectuar una medición sobre la capacidad de predicción del modelo generado a partir de un clasificador. Una alternativa para verificar o medir la bondad del clasificador es la matriz de confusión [31].

Matriz de confusión: Nos permite visualizar mediante una tabla de contingencia la distribución de errores cometidos por un clasificador. Esta matriz de confusión para el caso de las dos clases de nuestro proyecto tiene la siguiente apariencia, como se puede apreciar en la *Tabla 10*.

	CLASE PREDICHA	
CLASE REAL	Contenedor amarillo	No contenedor amarillo
Contenedor amarillo	Verdaderos positivos (VP)	Falsos negativos (FN)
No contenedor amarillo	Falsos positivos (FP)	Verdaderos negativos (VN)

Tabla 10: Matriz de confusión para dos clases

Se puede entender la *Tabla 10* si se entienden los siguientes términos:

- **Verdaderos positivos (VP)**: Instancias que van al contenedor amarillo y que el sistema reconoce que van a dicho contenedor.
- **Falsos negativos (FN):** Instancias que van al contenedor amarillo y que el sistema dice que no van ahí.
- **Falsos positivos (FP):** Instancias que no se depositan en el contenedor amarillo pero el sistema dice que sí que se depositan ahí.
- **Verdaderos negativos (VN):** Instancias que no van al contenedor amarillo y correctamente el sistema reconoce que no van ahí.

Suponiendo que N es el número del conjunto de datos de entrenamiento, entonces

$$N = VP + FN + FP + VN$$

Otros conceptos son los siguientes:

- **Tasa de error** = FP+FN/N
- **Tasa de acierto** = VP+VN/N
- **Precisión** = VP/(FP + VP)
- **Accuracy** = (VP + VN) / N

En este proyecto se escoge la *accuracy* porque ofrece una visión más general teniendo en cuenta todos los errores, mientras que el resto sólo tiene en cuenta parte de los errores.

Finalmente, se escogerá el clasificador que presente mejores indicadores. Estas métricas son también utilizadas en el proceso de *cross-validation*.

Cross-validation

Como ya se ha comentado, utilizar el método *cross-validation* permite el análisis estadístico para obtener otras medidas del rendimiento estimado, como la media y la varianza de una de las métricas anteriores.

En el caso de este proyecto, se evaluarán los algoritmos mediante un análisis estadístico que se explica a continuación. De lo que trata este análisis es de la comparación de los clasificadores. Para llevar a cabo esta comparación, se va usar la metodología descrita a continuación [32].

1. Se eligen los algoritmos que se desea comparar.
2. Se divide el banco de imágenes D en k subconjuntos $\{D_1,..., D_k\}$, siendo lo habitual k=10.
3. Ahora, para cada uno de los clasificadores, se ejecuta una validación cruzada de tal forma que:
   - Para cada uno de los k subconjuntos de D, se crea el conjunto de entrenamiento $T = D \setminus D_k$.
   - Se mide la accuracy en D_k.

4. De este modo, se obtiene k precisiones $\{A^i_1,...,A^i_k\}$ para cada uno de los clasificadores C_i. La precisión total del clasificador es el promedio de las k precisiones. Entre los algoritmos, estos k valores también dan una estimación de la varianza.

Una vez realizada esta metodología, se tienen que distinguir varios casos. Primero se compueban las condiciones de parametricidad de las *accuracies* $\{A^i_1,...,A^i_k\}$:

- Independencia
- Normalidad:
- Heterocedasticidad

En la *Tabla 11* se puede ver qué se debe hacer si se cumplen o si no se cumplen las condiciones de parametricidad. En ambas, deben considerarse dos casos: se comparan dos algoritmos o se comparan más de dos algoritmos. Estos dos casos pueden verse en la tabla.

	Se comparan dos algoritmos	Se comparan más de dos algoritmos
Se cumplen condiciones de parametricidad	Se aplica un T-test [33] el cual dice, de manera significativa, si una técnica es mejor que la otra.	Se aplica un test ANOVA [34], el cual dice si hay diferencias significativas entre los distintos clasificadores, pero no cuáles son estos clasificadores (para ello se aplica el post-test Bonferroni-Dunn [35]).
No se cumplen condiciones de parametricidad	Se aplica el test de Wilcoxon [36] el cual dice, de manera significativa, si una técnica es mejor que la otra.	Se aplica un test Friedman [37], el cual dice si hay diferencias significativas entre los distintos clasificadores, pero no cuáles son estos clasificadores (para ello se aplica el post-test Holm test [38]).

Tabla 11: Casos a considerar respecto al cumplimiento de las condiciones de parametricidad

Estos tests miden el efecto, pero no el tamaño de dicho efecto. Para medir el tamaño del efecto se utiliza el estadístico d de Cohen [39]. Este estadístico se debe interpretar de la siguiente manera:

- Si toma un valor entre 0.2 y 0.3, el tamaño del efecto es pequeño, pero aun así es probablemente significativo.
- Si toma un valor cercano a 0.5, el tamaño del efecto es medio, por lo que la diferencia entre los clasificadores es evidente.
- Si toma un valor mayor que 0.8, el tamaño del efecto es grande, por lo que los clasificadores claramente difieren.

4.2.6. Resumen

Como conclusión de este apartado, se ha decido lo siguiente:

- Las fotos que formarán el banco de imágenes serán realizadas por un grupo de diez personas de Ecoembes, con el teléfono móvil y serán enviadas por WhatsApp a un número concreto.
- Para realizar la división del dataset, se utilizará el método *cross-validation*.
- La extracción de descriptores se llevará a cabo mediante el modo *transfer learning*, realizando un análisis estadístico entre las redes Densenet, GoogleNet, Inception, Xception, Overfeat, Resnet, VGG16 y VGG19.
- Mediante el análisis estadístico, entre los algoritmos de clasificación KNN, Árboles de decisión, Extra Trees, Regresión logística, SVM y Random Forest, se escogerá cuál utilizar dependiendo de los resultados.
- Se realizará una evaluación mediante un análisis estadístico de comparación de clasificadores.

4.3. Diseño de la aplicación móvil

4.3.1. Diseño de alternativas para la integración del modelo

A la hora de integrar el modelo desarrollado en el primer paso, existen dos opciones:

1. Almacenar los datos de la aplicación sin hacer uso de ningún servicio externo, de tal manera que cuando un ciudadano se descargue la aplicación, el modelo ocupará memoria de su tarjeta SD o de su tarjeta telefónica.
2. Hacer uso de un servidor. Es decir, hacer uso de un servicio externo donde estén almacenados los datos. Así, de esta forma, cuando un ciudadano selecciona o saca una foto, la aplicación móvil hace uso de ese servidor al cual le envía la foto. Después, es el servidor el que le envía la respuesta de si va al contenedor amarillo o no.

En la *Tabla 12* se pueden observar las ventajas y desventajas de ambas opciones.

	Ventajas	Desventajas
Con servidor	-Acceso fácil -Almacenamiento ampliable -Almacenamiento externo	-Es necesario Internet -Se necesita hacer uso de diferentes servicios
Sin servidor	-No es necesario Internet -Alta velocidad de acceso a la información	-Almacenamiento interno (tarjeta del móvil o SD) -Memoria limitada -Retraso en el procesamiento de datos

Tabla 12: Comparativa entre app móvil con y sin servidor

El fin de la aplicación es que la utilice el ciudadano para reciclar mejor, y si la descarga y ve que ocupa mucho espacio, seguramente la desinstalará al poco tiempo. Por lo que hay que dar facilidades al ciudadano. Si el almacenamiento es externo y utiliza un servidor, el acceso es más fácil y ocupará menos en el teléfono móvil. Por lo tanto, para este proyecto, se ha decidido hacer la aplicación móvil con servidor.

4.3.2. Diseño de interfaz

Para el diseño de interfaz, se han propuesto dos alternativas cumpliendo los requisitos de la aplicación. Se debe recordar que en los requisitos se indicaba que debía ser un diseño básico e intuitivo, y contar con los logos de *The Circular Lab* y Ecoembes al mismo nivel.

Diseño 1

Para este diseño, se han utilizado tonos verdes que representan mejor a *The Circular Lab* y a Ecoembes. La pantalla inicial será la que aparece en la *Pantalla 1.1.* Aquí, se pueden observar dos botones: uno de la galería y el otro de una cámara. Al pinchar sobre el de galería, se podrá seleccionar una foto de la misma. Si se pincha sobre el icono de cámara, será posible sacar una foto.

Pantalla 1.1: Pantalla de inicio diseño 1

Pantalla 1.2: Pantalla de comprobación foto diseño 1

Tras sacar una foto, o seleccionar una de galería, se verá la *Pantalla 1.2*. Aquí se verá la fotografía seleccionada y dará la opción de “Comprobar”.

Una vez en la *Pantalla 1.2*, si se le da a “Comprobar”, se mostrará la *Pantalla 1.3.1*, si hay que depositar el producto en el contenedor amarillo, o la *Pantalla 1.3.2*, si no hay que depositarlo en dicho contenedor.

Pantalla 1.3.1: Pantalla respuesta “Sí” diseño 1

Pantalla 1.3.2: Pantalla respuesta “No” diseño 1

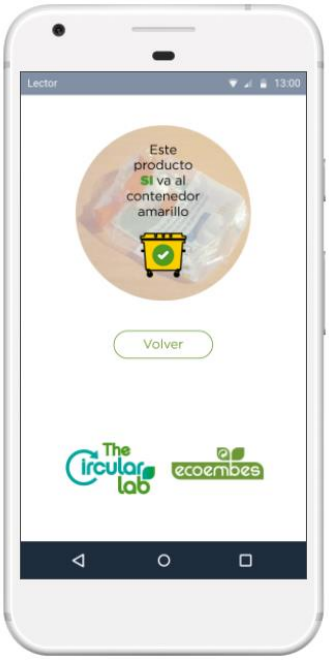
Diseño 2

Para este diseño, se han utilizado tonos blancos en el fondo y también el verde. La pantalla inicial será la que aparece en la *Pantalla 2.1.* Aquí, se pueden observar los botones de galería y de cámara como en el Diseño 1. Como en éste, al pinchar sobre el de galería, se podrá seleccionar una foto de la misma y si se pincha en el icono de cámara, sacar una foto.

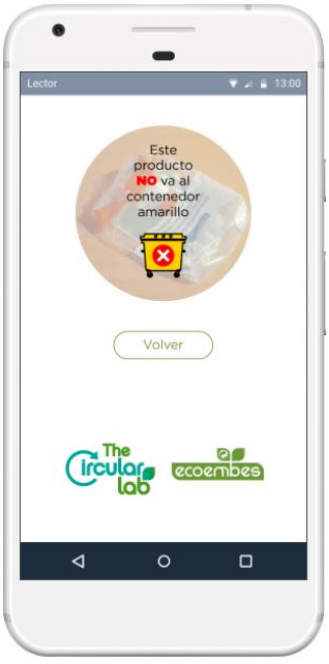
Pantalla 2.1: Pantalla de inicio diseño 2

Pantalla 2.2: Pantalla de comprobación foto diseño 2

Tras sacar una imagen, o seleccionarla de galería, la aplicación irá a la *Pantalla 2.2*, donde se verá la imagen seleccionada y se podrá "Comprobar". Si se le da a "Comprobar" en la *Pantalla 2.2*, se mostrarán o la *Pantalla 2.3.1* o la *Pantalla 2.3.2*, en función del mismo criterio que en el Diseño 1.



Pantalla 2.3.1: Pantalla respuesta "Sí" diseño 1



Pantalla 2.3.2: Pantalla respuesta "No" diseño 1

Han sido mostrados estos dos diseños al cliente. Le gusta más el Diseño 2 debido a que cree que el blanco le proporciona simplicidad. Por lo que se ha decidido escoger dicho diseño.

5. Implementación

Esta sección tiene dos apartados. En el primero se trata la parte correspondiente a la implementación del modelo, y en el segundo, la parte relacionada con la aplicación Android, la cual consta de cómo crear el servidor y la propia aplicación móvil.

5.1. Desarrollo del modelo

En esta sección, se enumeran las tecnologías necesarias para la construcción del modelo y se van explicando los pasos a seguir para lograrlo.

5.1.1. Tecnologías modelo

Para este trabajo, se ha utilizado Linux como sistema operativo debido a que facilita la instalación de las librerías necesarias, y se ha decidido utilizar las siguientes herramientas para generar el modelo:

- Python 3.5 [40] como lenguaje de desarrollo, ya que para problemas de clasificación de imágenes la mayoría de las librerías están disponibles en Python, por lo que es más conveniente.
- OpenCV 2.4.10 [41], que es una librería para visión por computador y aprendizaje automático.
- Scikit-learn [42], que es la principal librería que existe para trabajar con aprendizaje automático e incluye la implementación de un gran número de algoritmos de aprendizaje. Se puede utilizar para clasificadores, extracción de características, regresiones, agrupaciones, reducción de dimensiones, selección de modelos, o preprocesamiento.
- Pycharm Professional 2017.2.3 [43] como IDE de desarrollo para Python, ya que es muy completo y tiene una modalidad de edición libre. Además, es muy útil por su compatibilidad con múltiples marcos de desarrollo web de terceros como Django (que se verá en la sección *5.2.1. Desarrollo del servidor*).
- ObjectClassification[2], que es una librería de extracción de descriptores (utilizando *transfer learning*) y análisis estadístico desarrollada por el grupo de Informática de la Universidad de La Rioja.
- Google Drive [44] para el almacenamiento del banco de imágenes y de documentos relacionados con el TFG.

[2] La librería ObjectClassification se aloja en github.com/joheras/ObjectClassificationByTransferLearning

5.1.2. Construcción modelo

Análisis estadístico

Tras configurar el entorno sin ningún problema reseñable, se ha llevado a cabo un análisis estadístico para elegir la mejor combinación de extractor de descriptores y modelo de aprendizaje para nuestro problema concreto. Para ello, se ha utilizado un código perteneciente a la librería ObjectClassification en el que se ha configurado una serie de parámetros. Esta librería contiene, principalmente, la siguiente estructura de ficheros python:

```
I--- ObjectClassification
    I--- conf
        I--- conf.py
    I--- DensenNet
    I--- extractor
        I--- keras
        I--- sklearn_theano
    I--- indexer
    I--- output
        I--- resultados
    I--- pruebas
    I--- results
        I--- resultados
    I--- shallowmodels
    I--- sklearn
    I--- StatisticalAnalysis
    I--- utils
    I--- Comparing.py
    I--- h5py2csv.py
    I--- index_features.py
    I--- prediction.py
    I--- StatisticalComparison.py
    I--- train.py
```

Esta estructura sólo muestra los ficheros y directorios más relevantes y algunos de los que se mencionan posteriormente. A continuación, se explican los más relevantes para este trabajo.

- conf/conf.json: fichero donde se configuran, entre otros, los siguientes parámetros:
    - dataset_path: indica dónde está situado nuestro banco de imágenes
    - model: indica la red que se va a utilizar para extraer descriptores
    - modelClassifier: indica el algoritmo de clasificación que se va a utilizar

- output/resultados: en este directorio se almacenarán los resultados al extraer las características de los modelos.
- results/resultados: en este directorio se almacenarán los ficheros csv resultado del análisis estadístico.
- StatisticalComparison.py: en este fichero se comparan todos los algoritmos y se realiza el análisis estadístico.
- index_features.py: se utiliza para extraer las características del modelo.
- Comparing.py: este fichero se utiliza para comparar algoritmos.
- train.py: en este fichero se entrena el modelo.
- prediction.py: es el fichero que realiza la clasificación de la imagen y predice a qué clase va.

Para realizar el análisis estadístico, los pasos a seguir son los siguientes:

1. Construir el banco de imágenes y organizarlo en carpetas. Es decir, hay una carpeta principal llamada “proyecto”. Ésta contiene dos carpetas: “amarillo”, que es la que contiene las fotos del banco de imágenes que corresponden a productos que sí que van al contenedor amarillo, y “otrocontenedor”, que contiene el resto de imágenes, es decir, las de productos que se deben depositar en otro contenedor.
2. Extraer descriptores para cada modelo. Para ello, se siguen los pasos que se muestran a continuación.
   2.1. En el fichero conf.json de la carpeta conf de ObjectClassification, se modifica la variable “dataset_path” para indicar que el banco de imágenes creado, se encuentra en “/home/master/Escritorio/proyecto”.
   2.2. Extraer las características del banco de imágenes usando el comando

```
$python index_features.py -c conf/conf.json
```

   Se extraen las características para cada modelo, es decir, para las redes Densenet, GoogleNet, Inception, Xception, Overfeat, Resnet, VGG16 y VGG19, como se dice en la sección *4. Diseño* (concretamente en *4.1.3. Extracción de descriptores*). Para ello, es necesario cambiar la variable “model” del fichero conf/conf.json de ObjectClassification por “densenet”, “googlenet”, “inception”, “xception”, “overfeat”, “resnet”, “vgg16” y “vgg19”, respectivamente. Para cada red se genera un fichero hdf5, correspondiente a los descriptores de imágenes, y un fichero cpickle, relacionado con la codificación de la clase. Los ficheros hdf5 pueden ser de diferentes tipos y sólo pueden ser abiertos generalmente, con la aplicación con la que han sido generados. Por otra parte, los ficheros cpickle son datos persistentes que se almacenan en los directorios de caché.

   2.3. Se realiza el análisis estadístico comparando diferentes métodos. Se han comparado los algoritmos KNN, Árboles de decisión, Extra Trees, Regresión logística, SVM y Random Forest, como se ha decidido en la sección *4. Diseño* (en *4.1.4. Algoritmos de aprendizaje*). Se lleva a cabo mediante el comando

```
$python StatisticalComparison.py -c conf/conf.json
```

La *Tabla 13* muestra la media (y la desviación típica) para los diferentes modelos estudiados, que se obtiene al realizar el paso 2.3. El mejor resultado para cada método de extracción de características está en negrita.

Modelos/ Redes	KNN	SVM	RF	LR	GB	ET	Test Friedman	Post-hoc
Densenet	68.5(6.3)	43.0(6.0)	73.0(5.6)	72.0(8.4)	71.5(6.7)	**76.5(6.3)**	8.83***	ET>RF, LR,GB> KNN>SVM
Resnet	94.5(4.7)	**97.0(3.3)**	85.0(7.1)	95.5(4.7)	91.0(7.0)	93.5(7.1)	7.30***	SVM>LR, KNN> ET>GB>RF
Overfeat	79.0(10.7)	**89.5(4.7)**	81.0(10.5)	89.5(5.2)	80.5(8.2)	85.0(8.4)	6.93***	SVM,LR> ET>GB> KNN>RF
Googlenet	**94.5(5.7)**	93.5(5.9)	89.0(4.4)	92.5(6.0)	89.5(6.9)	93.0(5.1)	2.02***	KNN,SVM, ET,LR>RF> GB
Inception	85.0(7.4)	64.0(13.6)	83.5(6.7)	**94.5(3.5)**	85.0(7.4)	88.0(6.4)	12.44***	LR>ET>GB, KNN>RF> SVM
Xception	**95.5(4.7)**	89.0(5.8)	86.5(5.5)	94.0(5.4)	87.5(5.1)	93.5(3.9)	5.81***	KNN>ET, LR>SVM, GB,RF
Vgg16	82.5(10.5)	43.5(6.7)	83.5(7.1)	**93.5(5.5)**	81.0(8.9)	89.0(7.7)	15.57***	LR>ET>RF, KNN>GB> SVM
Vgg19	86.0(5.4)	43.5(6.7)	83.5(7.1)	**93.0(5.1)**	86.0(6.6)	89.5(8.2)	13.62***	LR>ET> KNN,GB,RF >SVM

Tabla 13: Comparativa de los resultados de los diferentes modelos

Una vez hecho el análisis, se han originado documentos csv con el resultado de cada modelo en cada iteración. Entonces, se ha creado otro documento csv llamado “fold-comparison-best.csv”, en el que se recogen los mejores resultados para cada método de extracción de características en una tabla. Es decir, cada fila refleja los resultados del mejor algoritmo de cada red. Esto se ha realizado para saber qué modelo es el mejor de todos (junto con la red). Para cada red, los resultados de su mejor modelo pueden observarse en la *Tabla 14* (de cada modelo / red). Lo que se muestra es, como en la *Tabla 13*, la media (y la desviación típica), además del ranking del promedio del test de Friedman ordenados. Están ordenados de mejor a peor precisión y el mejor resultado, que es el de la primera fila, también está en negrita.

Técnica	Precisión	Ranking del promedio del test de Friedman
Resnet - SVM	**97.0(3.3)**	6.45
Xception - KNN	95.5(4.7)	5.9
Inception - LR	94.5(3.5)	5.25
Googlenet - KNN	94.5(5.6)	5.1
Vgg16 - LR	93.5(5.5)	4.75
Vgg19 - LR	93.0(5.1)	4.55
Overfeat - SVM	89.5(4.7)	2.95
Densenet - ET	76.5(6.3)	1.05

Tabla 14: Comparación de los mejores resultados modelo/red

Al mejor resultado se le aplica el proceso de ajuste de p-valor y se analiza el tamaño de efecto que tiene con el resto de técnicas. En este caso, el mejor resultado es de Resnet – SVM. Los resultados de analizar esta técnica con el resto de ellas pueden verse en la *Tabla 15*.

Técnica	Z-valor	p-valor	Ajuste del p-valor	d de Cohen
Xception - KNN	0.50	0.62	0.65	0.35
Inception - LR	1.10	0.27	0.65	0.70
Googlenet - KNN	1.23	0.22	0.65	0.51
Vgg16 -LR	1.55	0.12	0.48	0.73
Vgg19 - LR	1.73	0.083	0.41	0.88
Overfeat - SVM	3.20	0.001	0.008	1.75
Densenet - ET	4.93	8.24×10^{-7}	5.77×10^{-6}	3.84

Tabla 15: Ajustes del p-valor con Holm y d de Cohen usando la técnica Resnet-SVM

Como se ha comentado, se ha comprobado que la mejor red es Resnet utilizando el modelo SVM. Por lo que se ha reconfigurado el fichero conf.json para que la variable "model" sea "resnet" y "modelClassifier" sea "SVM". Hay diferencias significativas entre ésta y la red Overfeat con SVM, ya que la media de ésta última es de 0.895, mientras que es de 0.97 la de Resnet con SVM. Pero, sin embargo, la diferencia más grande es con la red Densenet utilizando el modelo ET, ya que la media de ésta es de 0.765, la más pequeña de todas. Con las demás redes y sus respectivos modelos no hay diferencias significativas.

5.1.3. Entrenando modelo final

Una vez obtenido el modelo, se ha entrenado a través del comando

```
$python train.py -c conf/conf.json
```

y se ha obtenido un archivo cpickle, que es el que contiene el modelo. Para probar que funciona correctamente, se puede usar el comando

```
$python prediction.py -c conf/conf.json -i imagePath
```

donde imagePath es la ubicación en la que se encuentra la imagen. Este comando produce como resultado la predicción sobre la clase a la que pertenece el producto de la imagen, y dicho resultado es mostrado por línea de comandos.

5.2. Desarrollo de la aplicación móvil

En esta sección se explicará, por un lado, el desarrollo del servidor, y por otro, la implementación y las tecnologías escogidas para la construcción de la aplicación móvil. Además, cada apartado se acompaña de la explicación de los pasos que se van realizando.

5.2.1. Desarrollo del servidor

Para el desarrollo del servidor se ha utilizado Django [45], que es un framework web Python basado en patrón MVC [46] (modelo-vista-controlador) de alto nivel que fomenta un desarrollo seguro y un diseño limpio y pragmático. Además, es de código abierto y extremadamente escalable. El funcionamiento de este framework tiene estrecha relación con la asignatura *Programación de aplicaciones web (PAW)* impartida en el segundo cuatrimestre del cuarto año del Grado en Ingeniería Informática, ya que en esta asignatura se explica el patrón MVC.

Concretando más, podemos decir que, en realidad, Django sigue el patrón MTV (modelo-template-vista), en el que el modelo en Django sigue siendo modelo, la vista se llama template (plantilla) y el controlador se llama vista.

El primer paso ha sido configurar y ejecutar el entorno de desarrollo en el que se trabaja. Para ello se han necesitado tres paquetes:

- paquete NumPy, que es el paquete fundamental para la informática científica con Python. Además de sus usos científicos, también se puede usar como un contenedor multidimensional eficiente de datos genéricos.
- paquetes django, que contienen el framework web Django.
- paquete de *requests* (solicitudes), para hacer que la interfaz con la API web sea más manejable.

Después, se ha llevado a cabo la creación del proyecto Django llamado “amarillo_api”. Esto se ha realizado a través del comando

```
$django-admin startproject amarillo_api
```

El directorio “amarillo_api” ahora contiene todo el código y configuración necesarios para poder ejecutar el proyecto Django. La estructura de directorios de este nuevo proyecto es la siguiente:

```
I--- amarillo_api
    I--- amarillo_api
        I--- __init__.py
        I--- settings.py
        I--- urls.py
        I--- wsgi.py
    I--- manage.py
```

Un proyecto Django consiste en múltiples aplicaciones, por lo que tenemos que crear "aplicaciones" separadas dentro del proyecto "amarillo_api". En este caso, se ha creado una llamada "amarillo_classification", que alberga el código necesario para construir una API de clasificación de imágenes de productos en función de si van al contenedor amarillo o no. Para ello, se utiliza el comando

```
$python manage.py startapp amarillo_classification
```

una vez que estamos en "amarillo_api" en la consola. De este modo, se tiene el directorio "amarillo_classification" dentro del proyecto "amarillo_api", el cual pasa a tener la siguiente estructura:

```
I--- amarillo_api
    I--- amarillo_api
        I--- __init__.py
        I--- settings.py
        I--- urls.py
        I--- wsgi.py
    I--- amarillo_classification
        I--- __init__.py
        I--- admin.py
        I--- migrations
        I--- models.py
        I--- tests.py
        I--- views.py
    I--- manage.py
```

Tras crear "amarillo_classification", se ha modificado views.py, de tal manera que si se hace una petición POST y se le envía una imagen, se haga uso del clasificador creado en la sección anterior, *5.1.2. Construcción modelo,* para devolver la predicción de éste. Es decir, para devolver "amarillo" u "otrocontenedor". La respuesta está en formato JSON con los campos *success*, que puede ser true o false, y *prediction*, que es la predicción y por tanto, sólo puede ser "amarillo" u "otrocontenedor". La respuesta sólo contendrá el campo *prediction* cuando *success* sea true. Y esto sucederá cuando reciba una petición JSON y una imagen. Si no recibe tal petición o la recibe pero sin imagen, la respuesta sólo contendrá el campo *success*, siendo éste false.

Por último, se ha actualizado el archivo "amarillo_api/amarillo_api/urls.py" para incluir la url de la clasificación de imágenes. Antes de pasar al desarrollo de la aplicación Android, se ha probado que el servidor funcionaba de manera correcta usando POSTMAN (extensión de Google Chrome).

5.2.2. Tecnologías aplicación móvil

Para el desarrollo de la aplicación móvil, se ha escogido Android Studio, que es el IDE oficial de Android. Hoy en día, Android es uno de los sistemas operativos que lideran el mercado de la tecnología de la comunicación, y ya que la aplicación móvil a desarrollar es Android, se considera conveniente usar su IDE oficial para el desarrollo de aplicaciones. Android Studio está basado en IntelliJ IDEA, que es un entorno de desarrollo para programas, que posee una serie de herramientas de edición de código.

Las ventajas que posee Android Studio son las siguientes:

- Compilación rápida.
- Ejecución de la app a tiempo real gracias al emulador.
- Ejecución de la app directamente desde el móvil.
- Contiene todo lo necesario para desarrollar un buen IDE.
- Posee capacidad para asociar automáticamente carpetas y archivos con su papel en la aplicación o la creación de nuevas carpetas.

La desventaja que tiene es que los requisitos son un poco elevados, es decir, que es necesaria una buena máquina para que funcione bien el emulador. Pero justo esto, es lo que hace que sea el mejor entorno para programar en Android y por ello, se ha escogido Android Studio para desarrollar la aplicación móvil.

Antes de pasar al desarrollo de la aplicación móvil, es necesario instalar Android Studio, y como se ha comentado, se está trabajando en Linux, por lo que lo tenemos que instalar para este sistema operativo. Para ello, primero se necesita instalar el JDK de Java. Una vez instalado, se ha descargado el IDE desde la página principal de Android Studio[3]. El paquete descargado tiene de nombre "android-studio". Tras descargarlo, se descomprime, y tiene la siguiente estructura de directorios a primera vista:

```
I--- android_studio
    I--- bin
    I--- gradle
    I--- jre
    I--- lib
    I--- license
    I--- plugins
```

Dentro de la carpeta bin hay un archivo llamado studio.sh que sirve para arrancar el programa. Por lo que ahora, mediante la consola, accedemos a la carpeta android-stdio/bin y ejecutamos el script studio.sh mediante

```
$sh studio.sh
```

y de esta manera, se arranca Android Studio. Por último, siguiendo los pasos de la configuración inicial que aparece, se instala.

[3] La página oficial de Android Studio es https://developer.android.com/studio/index.html

5.2.3. Desarrollo aplicación móvil

Para desarrollar la aplicación móvil, se ha creado un proyecto en Android Studio llamado proyectoapp. Consta de una parte de presentación y otra de lógica.

- Presentación: Esta capa consta de los ficheros xml en el layout. Concretamente, son cuatro ficheros que corresponden con el diseño de lo decidido en la sección *4.3. Diseño de la aplicación móvil*. Es decir, un fichero xml llamado activity.xml corresponde con la pantalla inicial (*Pantalla 2.1*), otro llamado comprobar_foto.xml con la segunda (*Pantalla 2.2*), y los dos restantes, contenedor_amarillo.xml y otro_contenedor.xml a las dos últimas (*Pantalla 2.3.1* y *Pantalla 2.3.2* respectivamente).

- Lógica: La capa de lógica se ha desarrollado siguiendo dos funcionalidades:

    1. Mostrar los ficheros xml de la capa de presentación.
    2. Invocar al servidor mediante una petición POST. Cuando el usuario pulsa el botón "Comprobar" en la aplicación, la capa de lógica se encarga de realizar la petición. Como es necesario el path de la imagen para ello, se han desarrollado dos métodos (getPathGallery y getPathCamera), de tal manera que, a partir de la uri de la foto, devuelven el path (dependiendo si es una foto de galería o de la cámara). Por otro lado, también se gestiona no sólo que en la petición se envíe una imagen, sino que la respuesta se obtenga en formato JSON (que es como la devuelve Django)

Durante el desarrollo de la aplicación han ido surgiendo problemas. Uno de ellos ha sido que, al principio, a través de la uri de la imagen, no se podía conseguir el path con ningún método de ninguna librería. Por ello, se han creado los métodos getPathGallery y getPathCamera, nombrados anteriormente, con los cuales ha sido posible.

Otro problema ha residido en la conexión con el servidor. Al principio, no era posible la conexión entre Android Studio y Django, por lo que no funcionaba la petición POST. Para solucionarlo, en Django, en amarillo_api/amarillo_api/settings.py, en la variable "ALLOWED_HOSTS" ha sido necesario poner la IP correspondiente al servidor (no servía trabajar con localhost).

Se han realizado numerosas pruebas de la aplicación tanto en el emulador de Android Studio, como en móviles con versiones Android superiores a la 6.0, y en todas ellas la aplicación funciona correctamente. Por lo que podemos concluir que la aplicación móvil de Android está terminada y es completamente funcional.

6. Seguimiento y control

Para el seguimiento y control de este proyecto se ha utilizado la aplicación web de la Universidad de La Rioja. En ella es necesario insertar el día que empiezas y a partir de esta información, la aplicación genera cuatro puntos de control, que corresponden al mismo día de cada mes desde que se empieza.

6.1. Primer punto de control

El primer punto de control es el día 15 de octubre. En la *Figura 17* pueden verse las horas que verdaderamente llevaba trabajando en este proyecto (en azul), frente a las horas que se tenían programadas el día 23 de octubre (en rojo).

Figura 17: Comparación horas del proyecto reales (azul) y planificadas (rojo) 23 octubre

Como puede observarse en la *Figura 17*, prácticamente las horas invertidas en el proyecto hasta ese día, se corresponden con las previamente planificadas. Aunque, es cierto que, las reales sobrepasan a éstas últimas un poco (alrededor de seis horas). Además, en la *Figura 18* se puede apreciar el progreso de las distintas fases (planificación, análisis, diseño y producto) que ha habido hasta el día 23.

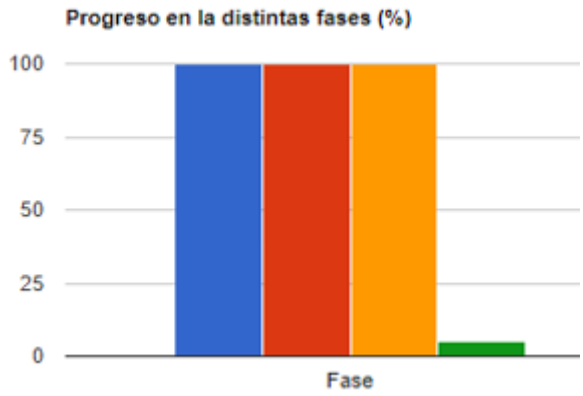


Figura 18: Evolución en las fases (de derecha a izquierda) planificación, análisis, diseño y producto, hasta el 23 octubre

6.2. Segundo punto de control

El segundo punto de control es el día 30 de noviembre. A partir aquí se decide modificar los puntos de control establecidos por la aplicación y realizar los gráficos de manera manual con las comparaciones entre las horas reales y las horas establecidas para dichos días. Ya que, de esta forma, se llevaba a cabo una mejor organización del proyecto. Las horas planificadas para el 30 de noviembre son 135, como puede verse en la *Figura 19*.

Figura 19: Comparación horas del proyecto reales (azul) y planificadas (rojo) 30 noviembre

Lo que muestra la *Figura 19* es que, pese a las diez horas de más que se llevaron a cabo en el primer punto de control, se ha conseguido un equilibrio entre las horas planificadas y las horas reales, ya que son las mismas. En este punto, la fase del producto alcanzó el 50%.

6.3. Tercer punto de control

El tercer punto de control es el día 20 de enero. Para tal fecha se habían planificado un total de 185 horas. Como puede observarse en la *Figura 20*, las fechas planificadas corresponden con las reales, por lo que se sigue el plan satisfactoriamente.

Figura 20: Comparación horas del proyecto reales (azul) y planificadas (rojo) 20 enero

Por otra parte, la fase del producto asciende al 80% de su totalidad.

6.4. Cuarto punto de control

El cuarto y último punto de control es el día 9 de febrero. Las horas programadas para ese día son 270 aproximadamente. Todo sigue en la misma línea, las horas reales se corresponden a las horas planificadas, como puede verse en la *Figura 21*.

Figura 21: Comparación horas del proyecto reales (azul) y planificadas (rojo) 9 febrero

La false del producto ya alcanza el 100%.

Tras la fecha del último punto de control, se ha perfeccionado el desarrollo del producto y se han realizado las pruebas procedentes para su comprobación. Por lo que puede decirse que se han llevado a cabo las horas planificadas, pero las pruebas realizadas al producto se han realizado más tarde de lo previsto. Pese a esto, las fases de planificación, análisis y diseño han sido terminadas según corresponde.

7. Conclusiones

7.1. Balance

Tras finalizar este proyecto, puedo decir que no sólo se han cumplido las expectativas iniciales, sino que el grado de satisfacción personal es mayor del imaginado, ya que mis conocimientos se han ampliado notoriamente. Se ha construido una aplicación móvil lista para ser desplegada con un acierto del 97% (muy superior al 75% que se había marcado como objetivo inicial). Tras construir el banco de imágenes, se logró realizar un análisis estadístico, lo que permitió conocer qué red junto a qué modelo daban los mejores resultados. Tras esto, se realizó el servidor en Django, pudiendo recibir éste una imagen y devolver una respuesta JSON. Después, en último lugar, se ha llevado a cabo el desarrollo de la aplicación Android, pudiendo hacer a través de ésta, una petición POST al servidor. En cuanto al número de horas planificadas con las que realmente he invertido, no hay una diferencia significativa, lo que es realmente bueno.

En cuanto a mi experiencia personal, al comenzar el proyecto, noté que la parte de diseño estaba ocupando mucho tiempo dentro de éste. Al terminarlo, puedo entender el porqué. Es una de las partes más importantes y tenerla tan bien acabada, me ha permitido evitar confusiones a la hora de la implementación.

La implementación quizás es donde más problemas han aparecido, ya que apenas tenía conocimientos acerca de Android Studio. Gracias a este proyecto, he adquirido mucha práctica. Por otro lado, un tema en el cual tenía mucho interés era la Inteligencia Artificial, ya que en las prácticas en Ecoembes comencé a sumergirme en ese mundo, y tenía ganas de seguir nadando en él. Este proyecto así me lo ha permitido. Durante las prácticas no pude trabajar mucho con Django, y ahora siento que he aprendido mucho, aunque sé que es un tema muy amplio y ni podría enumerar las cosas que me quedan por aprender. Aun así, no sólo me ha dejado aprendizaje, sino que también me ha impregnado de curiosidad por seguir aprendiendo, y eso es lo que considero más importante de todo. Además, claro está, de un trabajo hecho.

7.2. Lecciones aprendidas

Como lecciones aprendidas a lo largo de este proyecto, pueden destacarse las siguientes:

- Tener conocimientos de inglés, ya que la mayoría de la documentación que versa sobre Inteligencia Artificial y sobre Android está en este idioma.
- Siempre ver el lado positivo de los problemas que vayan surgiendo, y en lugar de dar paso a la frustración, aprender de ello, y buscar otras alternativas.
- Tener una comunicación fluida con el tutor, ya que a mí me ha ayudado mucho para aprender de los errores, ver las mejoras oportunas, y motivarme a buscar alternativas si en su caso así lo requería.
- Prever otras formas de implementar, ya que pueden surgir problemas y se puede perder mucho tiempo pensando en cómo hacerlo de otra manera.
- La organización es primordial y seguir el plan organizado dará los resultados que buscamos.

7.3. Mejoras

Por cuestiones de capacidad y tiempo, se decidió hacer el TFG de esta manera. Pero en realidad puede ampliarse de tal manera que sea de mayor ayuda para los usuarios a la hora de reciclar. Las mejoras pueden ser las siguientes:

- Más clases que “amarillo” y “otrocontenedor”. Este proyecto te dice si un producto va al contenedor amarillo o a otro contenedor. Pero en realidad, se puede ampliar de tal manera, que te devuelva siempre al contenedor que es. Es decir, no sólo el contenedor amarillo, sino el resto de los contenedores concretándolos.
- Más imágenes para entrenar el modelo. En este proyecto, debido a la capacidad de procesado del ordenador y al tiempo, se han seleccionado productos para entrenar al modelo, pero en realidad, se podría hacer con mucha más variedad de productos si se tuviera mucha más capacidad disponible.
- Desplegar la aplicación para su uso.
- Crear una aplicación móvil ligera que incluya el modelo, para evitar retardos ligados al envío de peticiones a un servidor.

8. Bibliografía

[1] «Separar es reciclar» 2016. [En línea]. Available: https://play.google.com/store/apps/details?id=appinventor.ai_legosanagus.SepararParaReciclar. [Último acceso: octubre 2017]

[2] «Guía del reciclaje» 2011. Enlace: https://play.google.com/store/apps/details?id=es.guiareciclaje.ecoembes&hl=es. [Último acceso: octubre 2017]

[3] «RecilcApp-BiziklApp» 2015. Enlace: https://play.google.com/store/apps/details?id=es.navarra.appreciclaje&hl=es. [Último acceso: octubre 2017]

[4] «Reciclar,» 2016. Enlace: https://play.google.com/store/apps/details?id=org.siga.reciclar&hl=es. [Último acceso: octubre 2017]

[5] «Aprende a reciclar» 2013. Enlace: https://play.google.com/store/apps/details?id=air.recycleGame&hl=es. [Último acceso: octubre 2017]

[6] «Redes neuronales,» 2013. Enlace: http://redesneuronares.blogspot.com.es/. [Último acceso: octubre 2017].

[7] University of Washington, «The History of Artificial Intelligence,» diciembre 2016. Enlace: https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf. [Último acceso: octubre 2017].

[8] Woodward y Neil, «No Free Lunch Theorem,» 2003. Enlace: http://www.no-free-lunch.org/.

[9] «Google,» Enlace: https://www.google.es. [Último acceso: noviembre 2017].

[10] «Imagenet,» Enlace: http://www.image-net.org/. [Último acceso: noviembre 2017].

[11] «Pexels,» Enlace: https://www.pexels.com/. [Último acceso: noviembre 2017].

[12] «Pixabay,» Enlace: https://pixabay.com/es/. [Último acceso: noviembre 2017].

[13] «Unplash,» Enlace: https://unsplash.com/. [Último acceso: noviembre 2017].

[14] Wilfrido Gómez Flores, «Análisis de datos. Validación de clasificadores.,» Enlace: http://www.tamps.cinvestav.mx/~wgomez/diapositivas/RP/Clase14.pdf. [Último acceso: noviembre 2017].

[15] O. Rodríguez, «Validación cruzada (cross-validation) y remuestreo (bootstrapping),» Enlace: http://www.oldemarrodriguez.com/yahoo_site_admin/assets/docs/Presentaci%C3%B3n_-_CV.293124233.pdf. [Último acceso: noviembre 2017].

[16] F. I. y. C. Saavedra, «Redes Neuronales Artificiales,» Enlace: http://vipo.uta.cl/charlas/volumen16/Indice/Ch-csaavedra.pdf. [Último acceso: noviembre 2017].

[17] Caglar Gulcehre, «Deep learning,» 2006. Enlace: http://deeplearning.net/. [Último acceso: noviembre 2017].

[18] Sebastian Ruder, «Transfer Learning,» Enlace: http://ruder.io/transfer-learning/. [Último acceso: noviembre 2017].

[19] News Center LATAM, «Microsoft,» 18 diciembre 2015. Enlace: https://blogs.technet.microsoft.com/microsoftlatam/2015/12/18/investigadores-de-microsoft-ganan-imagenet-el-desafo-de-visin-artificial/. [Último acceso: noviembre 2017].

[20] Gao Huang, Zhuang Liu, Laurens var der Maaten, Kilian Q. Weinberger, «Densely Connected Convolutional Networks,» Enlace: http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf. [Último acceso: noviembre 2017].

[21] Leonardo Araujo Santos, «GoogleNet and Inception Layer,» Enlace: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html. [Último acceso: noviembre 2017].

[22] François Chollet, «Xception: Deep Learning with Depthwise Separable Convolutions,» Enlace: https://arxiv.org/pdf/1610.02357.pdf. [Último acceso: noviembre 2017].

[23] Xiang Zhang, «OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,» Broadway, 2013.

[24] Adrian Rosebrock, «ImageNet: VGGNet, ResNet, Inception, and Xception with Keras,» 20 marzo 2017. Enlace: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/. [Último acceso: noviembre 2017].

[25] Miguel Cárdenas Montes, «KNN,» noviembre 2015. Enlace: http://wwwae.ciemat.es/~cardenas/docs/curso_MD/knn.pdf. [Último acceso: diciembre 2017].

[26] J. A. Alonso Jiménez, M. A. Gutiérrez Naranjo, F. J. Martín Mateos y J. L. Ruiz Reina, «Árboles de decisión,» 2002-3. Enlace: https://www.cs.us.es/~jalonso/cursos/ia2-02/temas/tema-7.pdf. [Último acceso: diciembre 2017].

[27] Alejandro Sánchez, «Extra trees,» 2016. Enlace: http://alejandrosanchezyali.blogspot.com.es/2016/01/algoritmo-del-gradiente-descendente-y.html. [Último acceso: diciembre 2017].

[28] Miguel Cárdenas Montes, «Random Forest,» Enlace: http://wwwae.ciemat.es/~cardenas/docs/lessons/RandomForest.pdf. [Último acceso: diciembre 2017].

[29] P. Larrañaga, I. Inza y A. Moujahid, «Regresión logística,» Enlace: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t7logistica. [Último acceso: diciembre 2017].

[30] F. Izaurieta y C. Saavedra, «SVM,» Enlace: http://www.uta.cl/charlas/volumen16/Indice/Ch-csaavedra.pdf. [Último acceso: diciembre 2017].

[31] Alexandre Savio, «Matrices de confusión y otros valores estadísticos,» Enlace: http://www.ehu.eus/ccwintco/index.php/Matrices_de_confusi%C3%B3n_y_otros_valores_estad%C3%ADsticos. [Último acceso: diciembre 2017].

[32] S. Garcia et al., "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," Information Sciences, vol. 180, pp. 2044–2064, 2010.

[33] «T Test (Student's T-Test),» 6 enero 2018. Enlace: http://www.statisticshowto.com/probability-and-statistics/t-test/. [Último acceso: diciembre 2017].

[34] Francisco Montes, «ANOVA,» Enlace: https://www.uv.es/montes/biomecanica2004/anova. [Último acceso: diciembre 2017].

[35] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2003.

[36] «Test de Wilcoxon,» Enlace: http://users.sussex.ac.uk/~grahamh/RM1web/WilcoxonHandoout2011.pdf. [Último acceso: diciembre 2017].

[37] Jaume LLopis Pérez, «Estadistica orquesta instrumento,» 14 noviembre 2013. Enlace: https://estadisticaorquestainstrumento.wordpress.com/2013/11/14/test-de-friedman/. [Último acceso: enero 2018].

[38] Hervé Abdi, «Holm test,» 2010. Enlace: https://www.utdallas.edu/~herve/abdi-Holm2010-pretty.pdf. [Último acceso: enero 2018].

[39] Juan Cruz Ripoll, «Comprensión lectora basada en evidencias,» 26 octubre 2011. Enlace: https://clbe.wordpress.com/2011/10/26/la-d-de-cohen-como-tamano-del-efecto/. [Último acceso: enero 2018].

[40] «Python,» Enlace: http://devdocs.io/python~3.5/. [Último acceso: enero 2018].

[41] «OpenCV,» Enlace: https://opencv.org/. [Último acceso: enero 2018].

[42] Scikit-learn developers, «Scikit-learn user guide,» 4 junio 2012. Enlace: http://www.math.unipd.it/~aiolli/corsi/1213/aa/user_guide-0.12-git.pdf. [Último acceso: enero 2018].

[43] «Pycharm,» Enlace: https://chocolatey.org/packages/Pycharm/2017.2.3. [Último acceso: enero 2018].

[44] «Google Drive,» Enlace: https://www.google.com/intl/es_ALL/drive/.

[45] «Django,» Enlace: https://www.djangoproject.com/. [Último acceso: enero 2018].

[46] Uriel Hernández, «Modelo-Vista-Controlador,» 22 octubre 2015. Enlace: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado. [Último acceso: febrero 2018].

A.1. Apéndice 1: Tipos aprendizaje

Aprendizaje no supervisado
Es un método de aprendizje automático donde un modelo es ajustado a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori. En el aprendizaje no supervisado, un conjunto de datos de objetos de entrada es tratado. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

Aprendizaje semisupervisado
Es una clase de técnicas de aprendizaje automático que utiliza datos de entrenamiento tanto etiquetados como no etiquetados: normalmente una pequeña cantidad de datos etiquetados junto a una gran cantidad de datos no etiquetados. El aprendizaje semisupervisado se encuentra entre el no supervisado y el supervisado. Los investigadores del campo del aprendizaje automático han descubierto que los datos no etiquetados, cuando se utilizan junto a una pequeña cantidad de datos etiquetados, pueden mejorar de forma considerable la exactitud del aprendizaje. La adquisición de datos etiquetados para resolver un problema suele requerir un agente humano capacitado para clasificar de forma manual los ejemplos de entrenamiento. El coste asociado al proceso de etiquetado puede hacer que un conjunto de entrenamiento totalmente etiquetado sea inviable, mientras que la adquisición de datos sin etiquetar es relativamente poco costosa. En estos casos, el aprendizaje semisupervisado puede ser muy útil.

Aprendizaje por refuerzo
Es un área del aprendizaje automático inspirada en la psicología conductista, cuya ocupación es determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de "recompensa" o premio acumulado. El problema, por su generalidad, se estudia en muchas otras disciplinas, como la teoría de juegos, teoría de control, investigación de operaciones, teoría de la información, la optimización basada en la simulación, estadísticas y algoritmos genéticos.

Transducción
Es similar al aprendizaje supervisado, pero su objetivo no es construir de forma explícita una función, sino únicamente tratar de predecir las categorías basándose en los ejemplos de entrada, sus respectivas categorías y los ejemplos nuevos al sistema.

Aprendizaje multi-tarea
Se fundamenta en la idea de que la tarea a resolver (tarea principal) se aprende conjuntamente con otras tareas relacionadas (tareas secundarias), se produce una transferencia de información entre ellas que puede ser ventajosa para el aprendizaje de la primera. Sin embargo, en problemas reales, es difícil encontrar tareas que estén relacionadas o incluso, encontrándolas, es sumamente complejo determinar el grado en que se va a realizar esa ayuda, ya que una tarea puede contener información que puede ayudar pero también perjudicar.

A.2. Apéndice 2: Métodos tradicionales de extracción de descriptores

1. Histogramas de color

Un histograma calcula la frecuencia de repetición de cada uno de sus valores dentro de un conjunto de datos. Por lo tanto, un histograma de color representa la distribución de color de una imagen. La forma más simple es, ante una imagen 2D en la que cada píxel representa un nivel de gris, calcular el histograma con varios niveles que corresponden a diferentes valores de gris. Este descriptor dará una representación genérica de la textura de la imagen, ya que muestra la distribución de la intensidad. Por ejemplo, en la *Figura A2.1* se puede observar un histograma de una imagen dividida en 250 niveles de gris. Se comprueba la presencia de muchos niveles oscuros (valores de intensidad menores de 100), por lo que se trata de una imagen oscura.

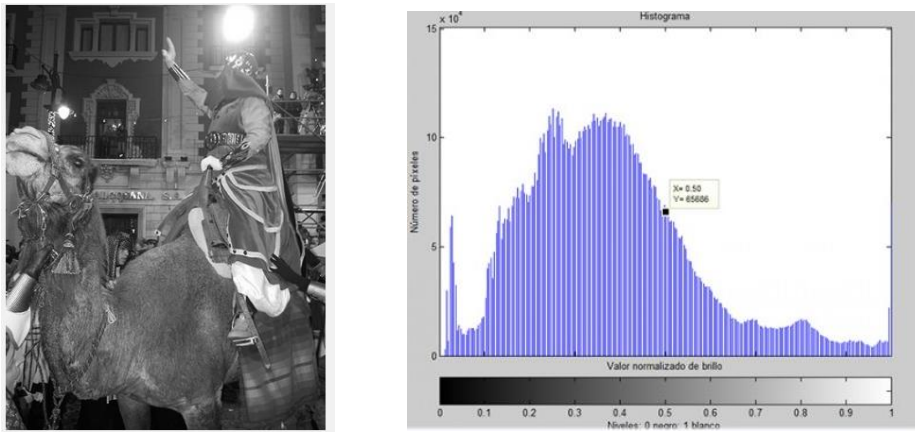


Figura A2.1: Imagen original y su histograma

Por lo que hay que asumir que estamos en un contexto donde las imágenes con similares distribuciones de color contienen contenidos visuales similares, lo que puede no ocurrir siempre.

Los histogramas de color son beneficiosos al representar cualquier imagen de cualquier tamaño usando un histograma de color 3D con 8 bins por canal, ya que el tamaño de la imagen no tendrá influencia en el tamaño del vector de descriptores final.

2. Histograma Haar [A1]

Las características de Haar derivan de un conjunto de filtros discretos y finitos, que se pueden aplicar computando su salida y acumulando ésta posteriormente en un histograma. La diferencia con otros métodos es que sus salidas no son complejas.

El wavelet de Haar en una dimensión fue propuesto en 1909 por Alfrèd Haar. Es también el wavelet más simple posible como se puede observar en la *Figura A2.2*. La desventaja técnica de este wavelet es que no es continuo y por lo tanto, no derivable.

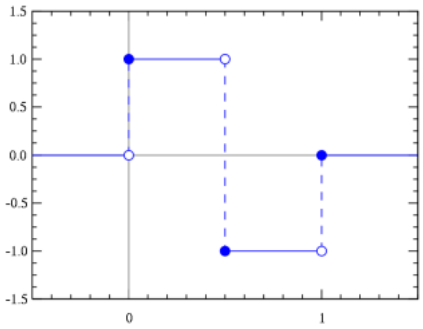


Figura A2.2: Wavelet de Haar en una dimensión

Este wavelet se propone como una función no diferenciable, de la siguiente forma:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & en\ otro\ caso \end{cases}$$

Si se extrapola este wavelet a dos dimensiones se obtienen los filtros de Haar, que permiten ser aplicados a las imágenes. Existen numerosas formas de aplicar estos filtros, los cuales se usaron por primera vez para la detección facial. En la *Figura A2.3* podemos ver uno de estos filtros.

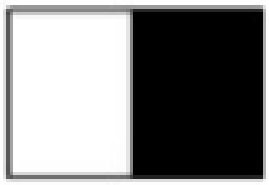


Figura A2.3: Un filtro de Haar 2D

En estos filtros, la sección negra indica un valor de -1 y la sección blanca un valor de +1, mientras que su extensión puede variar en función del objetivo de la aplicación. De esta forma ante una imagen, cada filtro generará un valor diferente que se acumulará en el descriptor final. En su modo más compacto, el histograma sólo tendrá dos valores que representarán el signo de la salida.

3. Histograma de gradientes orientados [A1]

El histograma de gradientes orientados (HOG) es un descriptor de la imagen que utiliza en cada uno de los píxeles, el gradiente como información básica.

El gradiente se define como un cambio de intensidad de la imagen en aquella dirección en la que el cambio de intensidad es máximo. Se calcula para cada uno de los píxeles de la imagen, Queda así definido para cada píxel los dos valores:

- La dirección donde el cambio de intensidad es máximo.
- La magnitud del cambio en esa dirección.

De esta forma, para cada píxel, estos dos valores nos permiten distinguir distintas situaciones de la configuración local alrededor del píxel en relación al cambio de contraste y a la forma local.

El cálculo del gradiente se puede realizar de varias formas diferentes. En el contexto del descriptor HOG este cálculo se realiza a partir de la diferencia de intensidad de los píxeles vecinos en dirección tanto horizontal como vertical.

Ahora bien, la información del gradiente local a nivel de cada uno de los pixeles se puede agregar en forma de histogramas calculados en diferentes áreas de la imagen. Por lo que podemos combinar estos histogramas para obtener el descriptor HOG como representación única y global de toda la imagen, que podremos utilizar en tareas de clasificación y detección de objetos.
Por una parte, el gradiente nos permite obtener información valiosa sobre la forma de un objeto, ya que generalmente los valores altos en la magnitud del gradiente se corresponden con el contorno o con la silueta de los objetos. Mientras que por otra, la orientación nos proporciona información sobre la forma del contorno de una forma invariante a la localización del objeto en la imagen.
El problema con esta representación a nivel de píxel del gradiente es que difícilmente se puede utilizar junto con un clasificador en un sistema de detección de objetos. Esto es debido a que un clasificador requiere normalmente como entrada una representación global de toda la imagen de una dimensión fija. Además, esta información a nivel de píxel puede ser muy sensible a pequeñas variaciones tanto en la forma como en la localización del objeto en la imagen. Ahora hay que convertir esta información local del gradiente para cada uno de los píxeles en una representación global de toda la imagen, en forma de vector de características que capture la forma global del objeto. Este descriptor global es el HOG. Para conseguir esta descripción global, el HOG actúa en dos pasos:

Divide la imagen en un número fijo de celdas, y para cada una de estas celdas se obtiene un histograma de las orientaciones de los gradientes de esas celdas. Para todas las celdas, se calculan los histogramas, combinándose todos estos histogramas para obtener la representación global de toda la imagen en forma de vector de características. A partir de estos pasos, cada uno de los histogramas de orientación permite capturar información de las orientaciones dominantes de la imagen, que se corresponderán con los valores altos en cada uno de los histogramas. Podemos ver un ejemplo de este método en la *Figura A2.4*.

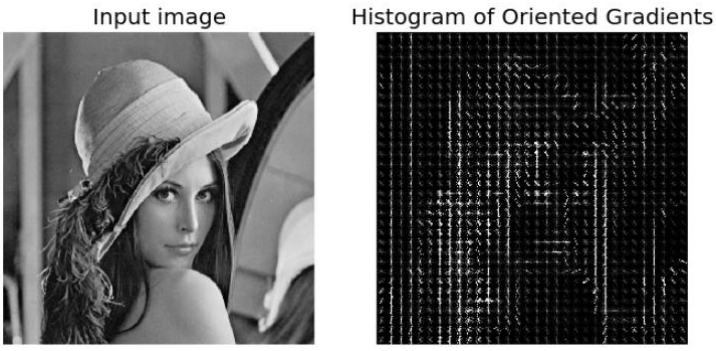


Figura A2.4: Ejemplo HOG

El punto de partida para obtener la representación final del descriptor son los histogramas de orientación del gradiente (calculadas para cada celda de la imagen). Una configuración bastante habitual en el descriptor HOG es utilizar bloques de dos celdas en horizontal y dos celdas en vertical. Un **bloque** es una agrupación de varias celdas vecinas. Se obtiene el vector de representación del bloque concatenando los histogramas de cada una de las celdas del bloque. Después, se normaliza este vector. La representación final del descriptor HOG se obtiene concatenando la representación normalizada de todos estos bloques.

4. Referencias

[A1] «Histogramas Haar y HOG» Enlace: https://es.coursera.org/learn/deteccion-objetos/lecture/GBJYS/l4-3-hog-calculo-de-los-histogramas. [Último acceso: diciembre 2017].